

**Role of the Best Practices from Extant Literature in Current Algorithm and Data
Structure Visualizations**

by

Ravali Gondi

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 06, 2011

Key words: Algorithm and Data Structure Visualization, Algorithm Animation, Best Practices,
Visualization Systems, Effect on Student's Learning, Teaching,
Design Requirements, Literature, Pedagogy

Copyright 2011 by Ravali Gondi

Approved by

Dean Hendrix, Chair, Associate Professor of Computer Science and Software Engineering
David Umphress, Associate Professor of Computer Science and Software Engineering
James Cross, Professor of Computer Science and Software Engineering

Abstract

Although there are many algorithm visualizations today, there is a question as to why algorithm and data structure visualization technology has not proved its effectiveness and gained widespread acceptance in mainstream computing education. While there are many likely reasons behind this, one possibility is that current visualization systems as a whole need to better focus on adopting best practices advocated in the research literature. In particular, this work conjectures that visualization systems need to better address certain pedagogical requirements and best practice features to be effectively used for education purposes. The widespread adoption of commonly accepted best practices would be seen as an important step in the maturation of the software visualization field. Indeed, only when software visualization systems that support an accepted, pedagogically effective feature set are common and widely accessible will these systems have the opportunity to make a significant impact on computing education.

After an extensive review of the extant literature, a set of best practices was selected to evaluate the central thesis question: To what extent do commonly available software visualization systems provide an appropriate, pedagogically effective feature set? This thesis question is further refined by four related research questions. A qualitative and quantitative analysis of data collected from the AlgoViz portal suggests that while a majority of visualization systems do not adhere to most best practices, there is a subset of relatively mature systems that provide a rich pedagogically effective feature set that is likely to enhance the teaching and learning environment.

Acknowledgments

I take pleasure in thanking many people for making my thesis possible. Firstly, I convey deep sense of gratitude to my major professor, Dr. Dean Hendrix for his extreme support, guidance, motivation, patience and importantly time on every step throughout my thesis work while understanding my situation always. Without his great help and suggestions, this thesis wouldn't have taken a successful shape. My sincere thanks also extend to the committee members, Dr. David Umphress and Dr. James Cross for being part of my advisory committee and their cooperation, proof reading and timely mentoring during my thesis. My heart filled thanks for my best sister, Rajitha Gondi and friends ShivaKrishna Shagabandi, Ramesh Bokka and Karthik Vemula for their ever helping hands, moral support and advice. I am truly grateful to GOD for giving me good opportunities that are tuning up my life as expected and for the adorable parents who have always been for me with their support, blessings and encouragement.

Table of Contents

Abstract	ii
Acknowledgments	iii
List of Tables	vii
List of Figures	viii
Chapter 1 Introduction	1
1.1 The motivation for algorithmic visualization	1
1.2 History of algorithm and data structure visualizations	5
1.3 Current use of visualizations.....	6
1.4 Different kinds of algorithm visualization users and their roles.....	7
1.5 Thesis question and approach for the analysis.....	10
Chapter 2 Definitions and Theory of Concepts	12
2.1 Software Visualization.....	12
2.1.1 Algorithm Visualization.....	13
2.1.2 Program Visualization	14
2.2 Algorithmic animation as a subfield of software visualization	14
Chapter 3 Literature Review	16
3.1 Design of algorithm visualizations	16
3.2 Analysis of requirements for algorithm visualization development from literature	17
3.2.1 Users' Analysis	17

3.2.2 Needs Analysis.....	18
3.2.3 Task Analysis.....	19
3.2.4 Information Analysis	20
3.2.5 Scope Analysis.....	21
3.2.6 Resources Analysis	22
3.3 Academic experience with algorithmic animation	23
Chapter 4 Best Practices for Effective Visualizations	26
4.1 Need to support flexible execution control of the algorithm visualization.....	27
4.2 Visualizations prompt users for answers to questions, predictions etc.....	29
4.3 Provide a context for users to interpret the visualization.....	31
4.4 Provide multiple views or representations	33
4.5 Allow user-specified data sets	35
4.6 Provide underlying source code and program execution.....	37
4.7 Can it be used as a lecture aid?	40
4.8 Can it be used for self-study?.....	42
4.9 Can it be used as a debugging aid?	43
Chapter 5 Different Visualization Systems.....	44
5.1 ANIMAL.....	44
5.2 JHAVE.....	47
5.3 JGRASP	49
5.4 JELIOT	53
5.5 MATRIXPRO	56
Chapter 6 Problem Statement	59

6.1 Motivation and associated research questions59
6.2 Methodology60
Chapter 7 Data Collection, Analysis, and Results62
7.1 Use of Algoviz.org.....	.62
7.2 Summary of the data collection process68
7.2.1 Data collection process69
7.3 Summary of the data collected.....	.71
7.4 Data analysis73
7.4.1 Data analysis to address the hypothesis74
7.4.2 Data analysis to address RQ176
7.4.3 Data analysis to address RQ278
7.4.4 Data analysis to address RQ380
7.4.5 Data analysis to address RQ484
Chapter 8 Summary and Conclusions.....	.87
References.....	.89
Appendix101

List of Tables

Table 7.1	Representation of the data collected based on the AlgoViz portal	72
Table 7.2	Representing best practices as stated in this thesis	73
Table 7.3	Data analysis to address the hypothesis	75
Table 7.4	Representation of the data collected based on the AlgoViz portal	77
Table 7.5	Data analysis to address research related question 1	77
Table 7.6	Representing best practices as stated in this thesis	77
Table 7.7	Data analysis to address research related question 2	78
Table 7.8	Representing best practices as stated in this thesis	79
Table 7.9	Data showing SV systems supporting more practices in order to address research related question 3	80
Table 7.10	Data showing SV systems supporting fewer practices in order to address research related question 3	82
Table 7.11	Data analysis to address research related question 4	85
Table 7.12	Representing best practices as stated in this thesis	86

List of Figures

Figure 1.1a	Proof without words: the sum of successive odd integers is the square of an integer	4
Figure 1.1b	Proof without words: bottom-up heap construction is $O(n)$	4
Figure 1.2	History of common Algorithm Visualization Systems	6
Figure 1.3	Schematic view of user roles	8
Figure 1.4	Software Visualization production process showing the roles involved.....	9
Figure 2.1	The classification of Software Visualization.....	12
Figure 2.2	Visualizing Software Visualization	14
Figure 3.1	Graphical representation of information in Quadtree Compression Algorithm.....	21
Figure 4.1	ANIMAL's animation display control tool bar	28
Figure 4.2	ANIMAL's speed control tool bar	28
Figure 4.3	Representation of dynamic questions during algorithm execution from JHAVE.....	30
Figure 4.4	Representation of textual description for easy interpretation of the visualization	32
Figure 4.5	Representing multiple views	34
Figure 4.6	Allowing the user to input any data set to explore algorithm behavior.....	36
Figure 4.7	Representation of program visualization and code highlighting	38
Figure 4.8	Representation of the view of call stack during a program execution	39
Figure 5.1	Algorithm animation using ANIMAL.....	46

Figure 5.2	Representation of algorithm execution from JHAVE	47
Figure 5.3	JHAVE representing algorithm visualization with pop-up questions.....	49
Figure 5.4	Simple representation of jGRASP tool.....	50
Figure 5.5	Representation of jGRASP features	52
Figure 5.6	Representation of Jeliot System	54
Figure 5.7	Representation of JELIOT 2000 System with more views	55
Figure 5.8	Red-Black Tree illustration in MatrixPro.....	57
Figure 7.1	Representation of recommended rating for a given visualization	63
Figure 7.2	Representation of AlgoViz Catalog.....	64
Figure 7.3	Representation of fields for a visualization entry.....	65
Figure 7.4	Field Reports entry in the AlgoViz portal	66
Figure 7.5	Representation of bibliography collection in AlgoViz portal	67
Figure 7.6	Snapshot of Forum discussions in AlgoViz portal.....	68
Figure 7.7	Description of a particular visualization in AlgoViz.....	70
Figure 7.8	Recommendation rating based on various categories for a visualization entry	70
Figure 7.9	Overall recommendation rating for a visualization entry.....	71
Figure 7.10	Data analysis to address research related question 3.....	83

CHAPTER 1

INTRODUCTION

Algorithms and data structures play a major role throughout computing education, and are the primary tools in software development. However, understanding these crucial concepts seems to be difficult for many students. From anecdotal evidence and general reports in the literature, many students in early computing courses find it very challenging to understand the concepts of algorithms and data structures, possibly because of the necessity to understand them both as high level abstractions and as concrete implementations.

Algorithm Visualization Systems are tools used to develop animations, graphical images, and other visualizations to illustrate the dynamic behavior of algorithms and data structures. These graphic representations are being used for designing, learning, explaining, and analyzing algorithms, as well as documenting and debugging programs. Algorithm visualization systems can be considered as a modern e-learning mechanism that aids students and teachers to better learn and teach both the abstract and detailed behavior of algorithms. Such systems, even if well designed, have very little educational value if they do not engage learners in pedagogically effective ways.

1.1 The motivation for algorithm visualization

An algorithm can be defined as the transformation of input data into output data with a precise sequence of computation that makes reference to a detailed data model. The fundamental

characteristic of any algorithm is that of computations that determine a specific trajectory in the state space of machine configurations [55]. In imperative programming languages, instances of this “state space of machine configurations” are modeled by single variables, arrays of values, linked lists, trees, and other data structures. Thus, in order to understand a program and its underlying algorithm, one must be able to mentally construct (i.e., *visualize*) a sequence of transformations effected by the computations on these data structures. It seems intuitively clear that an appropriate visualization of these transformations would make understanding more efficient.

Certainly the readability of an algorithm or program will make a difference in understanding. It is believed that visual animation of an algorithm or program will be much more helpful in understanding the process and flow of the algorithm or program compared to static reading techniques. According to [56], visual representations are also “data structures for expressing knowledge”. Even many psychologists believe that thinking needs visual images [92], defining the concept of “visual literacy” [57] that can be developed and trained in order to achieve better results.

Accepting the premise that software visualizations can play an important role in learning, an appropriate model with specific quasi-physical representations must be selected. To select a model, which will determine the particular representations and images that will be depicted, it is necessary to consider the appropriate physical animations and the appropriate physical objects they act upon. For example, in order for a visualization of a linked list insertion algorithm to achieve its intended pedagogical results, the model on which the visualization is based (say, boxes and arrows with fluid horizontal and vertical motion) must be able to be intuitively understood by the user and be directly indicative of the abstract objects and operations being

depicted (inserting a new node in a linked list of nodes). Many view the model as the single-most important determinate of visualization's success [54]. Consequently, the two important features of a successful algorithm animation are a suitable physical model for the abstract data types which are involved in the various operations and the granularity level at which the aggregation of the algorithm's steps will be defined [54]. The granularity level determines whether the model is applicable for both the high and low-level operations like the smallest changes that occur in each step of the algorithm. As a general goal, it is important to support both high and low-level granularities so that the more applicable one can be chosen by the user based on individual needs.

A long-standing tradition in mathematics called "proofs without words" provides a good example of effectively selecting a model and level of granularity. The intent of a proof without words is to clearly explain and justify the theoretical logic or soundness of a proposition without any verbal proofs [93]. That is, given a statement of a theorem, the picture alone is enough for a student to understand the truth of the theorem. For instance, consider a sequence consisting of the sum of consecutive odd numbers: 1, 1+3, 1+3+5, 1+3+5+7, and so on. A concept for students to understand is that the sum of n consecutive odd integers is a perfect square. A proof without words of this concept is shown in Figure 1.1a [54]. These kind of static pictures are helpful in mathematics to understand the premise without any verbal explanation.

Goodrich and Tamassia [58] have incorporated similar visual proofs without much explanation in the subject area of data structures and they have noticed that best way of recollecting the properties and characteristics of the data structures by the students is achievable if each property is associated with a suitable picture. For example, in order to prove that a heap

can be built in linear time from an unordered array they have used the single picture shown in Figure 1.1b [58].

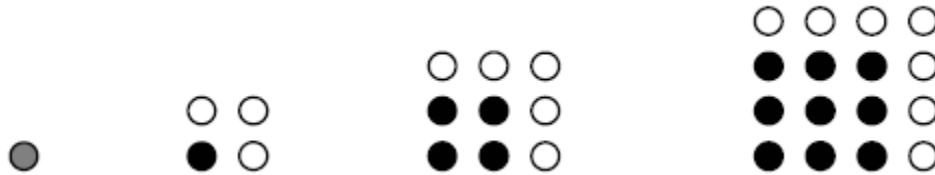


Figure 1.1a: Proof without words: the sum of successive odd integers is a perfect square [54].

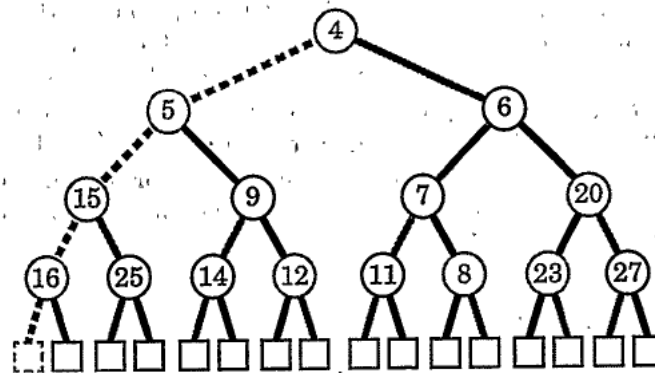


Figure 1.1b: Proof without words: bottom-up heap construction is $O(n)$ [58]

While Goodrich and Tamassia [58] have used these static pictures to help students learn certain *properties* of data structures and algorithms, static pictures with little other information are probably not as effective in helping students learn the *behavior* of data structures and algorithms. The complete behavior of an algorithm or program cannot be fully explained with still pictures. Thus, it is important to have a dynamic model with specific moving images (animations) explaining the process without any words.

1.2 History of algorithm and data structure visualizations

Algorithm and data structure visualizations have been playing important role for quite some time in the field of computing and computing education. The objective of Algorithm Visualization (AV) techniques is to enhance the understanding capability of students with regard to how the algorithms work by representing them graphically. This technology evolved in mid 1970's when instructors developed prototype systems that could produce animated films that represented the execution of programs [1]. The authors claimed that such systems would enable others to “produce short quick-and-dirty single-concept film clips with only hours of effort”[1]. This practice became more familiar through the well-known film visualization “Sorting out Sorting” developed by Ronald Baeker in 1981 [94]. The first well recognized visualization system for developing algorithm animations was Balsa [15], introduced in 1984. Other highly interactive systems quickly followed (e.g. [2, 3]), and then evolved into interactive programming environments in which users could develop their own visualizations (e.g. [4, 5]).

Many algorithm visualizations and visualization systems have emerged since then and are freely available to the users. This technology has had lot of growth over the past three decades, and Figure 1.2 shows a timeline for various algorithm animation systems developed during this period.

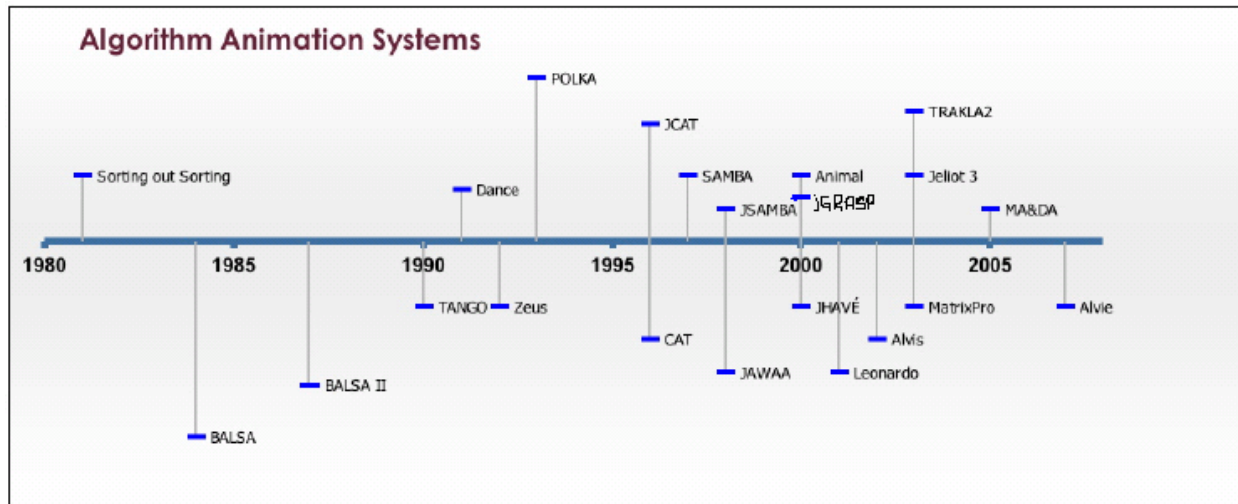


Figure 1.2: History of common Algorithm Visualization Systems from [34]

1.3 Current use of visualizations

Many visualization tools and systems have been developed to deliver “canned” animations as well as allow developers to construct their own algorithm animations. Such technology is now widely used to:

- Facilitate instructors explaining algorithm executions in a lecture (e.g. [2]);
- Help students use it themselves for improving their knowledge on the operations of various algorithms (e.g. [6]);
- Help instructors find errors in student programs while evaluating (e.g. [7]) and
- Educate students about basic understanding and operations of an abstract data type while working in a computer science laboratory (e.g. [8]).

It is intuitively accepted that visualization technology can serve as an effective alternative or supplement to written explanations in textbooks and verbal explanations in lectures. However, research indicates that there exist controversies on the efficacy of visualizations in practice. Some studies have shown that there is no significant difference in visualization

approach when compared to traditional learning [17, 19, 20], whereas other studies have shown that the algorithm and data structure visualizations can indeed enhance the learning of basic data structures that are typically part of a computer science curriculum [21, 18, 16]. Owing to the intuitive appeal of algorithm animation, there are many algorithm visualizations developed for most standard topics in CS 1, CS 2, and CS 3, but reflecting the reality of the research results, many of them do not have effective pedagogical value [27, 28, 14, 11]. To address this dichotomy, there has been research focused on identifying features that influence the effectiveness of software visualizations [22, 23, 25, 26, 29].

1.4 Different kinds of algorithm visualization users and their roles

Algorithm visualization is used by a wide variety of audiences, such as [12]:

- 1) *Researchers* performing research in the areas of software visualization or algorithm animation.
- 2) *Visualization Tool Developers* developing various visualization tools for visualizing algorithms, data structures and program execution graphically. Some of these tools directly provide visualizations for its users, such as *Jeliot 2000* [10], and a few serve as meta-tools that help the visualization developers to design their own visualizations, such as JAWAA [9] or ANIMAL [13].
- 3) *Visualization Designers* implement their visual representations by relating to the abstract theoretical concepts to help the students or learners understand the concepts in a more accurate manner.

- 4) *Students or Learners* use the visualizations to enhance their conceptual knowledge either by viewing or interacting with them using one or more engagement strategies described in [11].

These roles can be inter-changeable among the individuals. For example, if instructors self-build their own visualizations, they would take the visualization designers role.

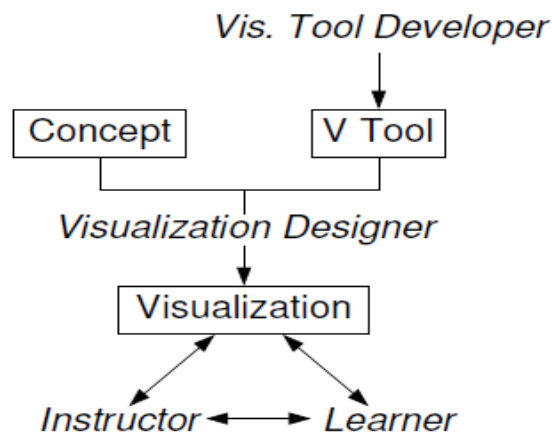


Figure 1.3: Schematic view of user roles [14]

Figure 1.4 [34] below represents another taxonomy of various persons that use software visualization techniques. Karavirta defines the roles as follows. Programmer is a person who is involved in developing program or algorithm and a SV system developer develops the software required to run the visualizations. Visualizer is the one who creates the visualizations and the user is the person who uses them. In practice, all these roles often overlap and a particular person can also take two different roles at the same time [30].

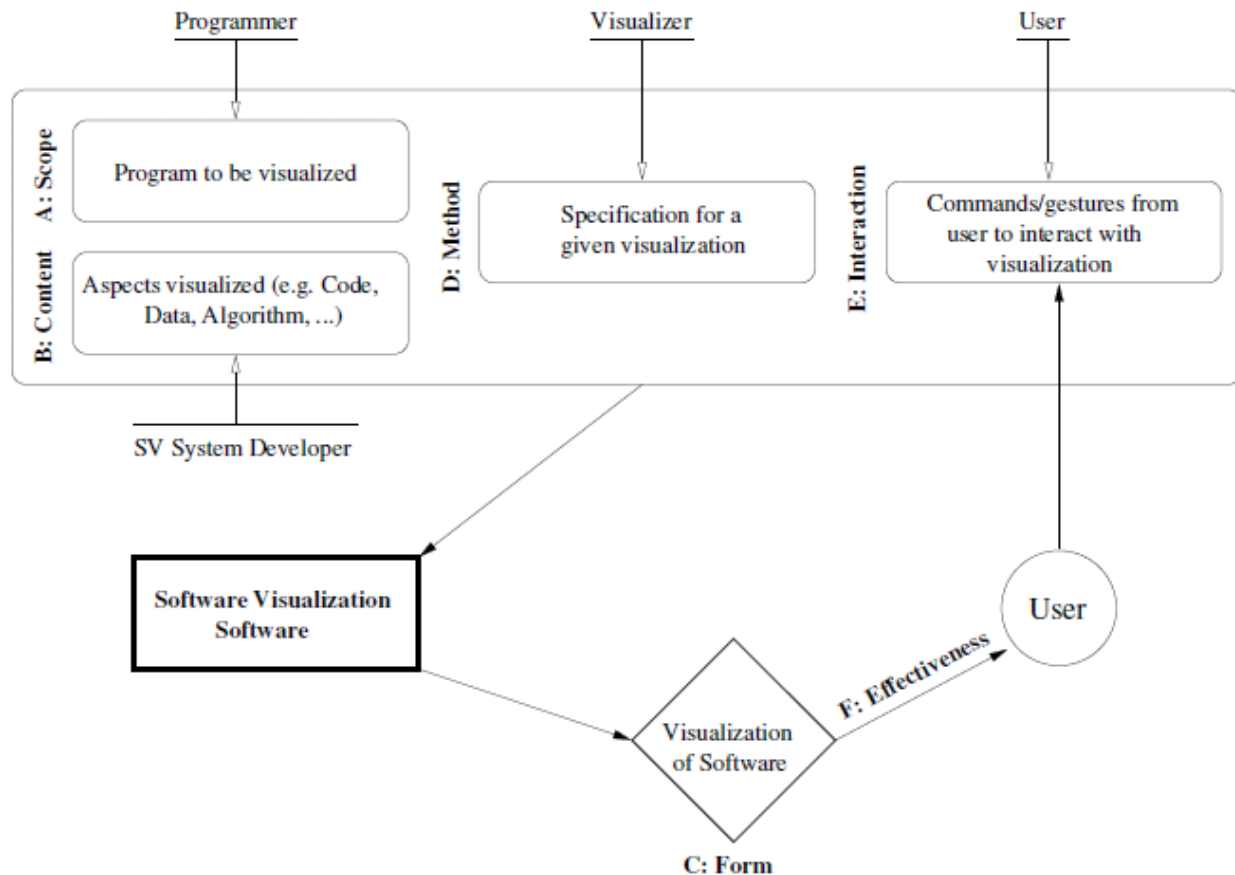


Figure 1.4: Software Visualization production process showing the roles involved [34]

As just discussed, the above two figures show the different roles of individuals and how they can interact with each other. Each of these roles has its own importance and expectations of visualizations. The visualization tool developers or SV system developers try to optimize their tools to produce better visualizations and try to meet the best practices advocated in the extant literature. On the other side, visualization designers or visualizers strive to use these tools and develop efficient visualizations that are useful to a large group of audiences. The instructors try to use these developed visualizations to incorporate them in their course material to enhance student learning and understanding capability and also to satisfy their teaching approach. And the learners hopefully learn the algorithm and data structure concepts better using these

visualizations. At times, instructors or learners adopt the role of visualization designer when trying to develop their own visualizations. There is a large set of users using these visualizations in different ways for which it is required to know how far each of these visualizations adhere to the best practices from the literature. This helps the instructors and students to better evaluate required visualizations before they can be used and also motivates the visualization tool designers and visualization designers to develop better products estimating its effective impact on student's learning.

1.5 Thesis question and approach for the analysis

After learning more about algorithm visualization, its technology and the effect it can have on learning, we reviewed the literature to identify a core set of features and characteristics that have been demonstrated to contribute to visualization's effectiveness. We will refer to these core features as "best practices." The primary question that my research addresses is the extent to which current, commonly available algorithm and data structure visualization systems adhere to these best practices advocated in the extant literature. The hypothesis that my research investigates is:

Hypothesis: *Most current visualization systems do not support most best practices.*

The primary thesis question and the research hypothesis are supported by four specific research questions.

RQ1: *To what extent do current visualization systems adhere to the best practices?*

RQ2: *Which best practices are most commonly supported by current visualization systems?*

RQ3: *Is there a set of visualization systems that, as a group, adhere to more best practices than visualization systems in general?*

RQ4: *What portion of visualization systems are recommended for use in the classroom?*

CHAPTER 2

DEFINITIONS AND THEORY OF CONCEPTS

2.1 Software Visualization

Price et al. [30] have defined software visualization as "the use of the crafts of typography, graphic design, animation, and cinematography with modern human computer interaction technology to facilitate both the human understanding and effective use of computer software." Software visualization is categorized into two parts – Algorithm Visualization and Program Visualization. Algorithm visualization is further categorized into static and dynamic algorithm visualization. The dynamic interactive graphic representation is same as Algorithm Animation, which is one form of software visualization, as shown in Figure 2.1.

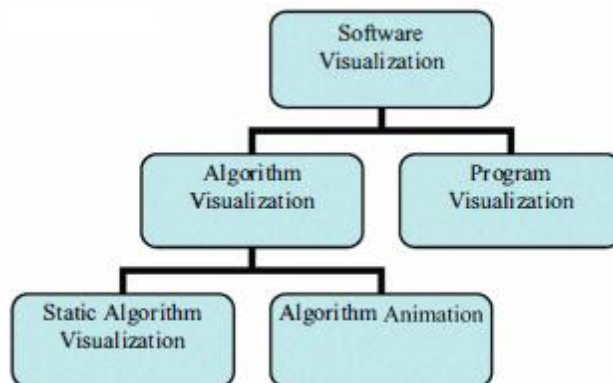


Figure 2.1: The classification of Software Visualization [31]

Software itself is very complex, abstract, creative, and difficult to observe. To better understand software, software visualization techniques use visual representations of complex software, information, algorithms, and data structures. To categorize them individually based on various themes within visualization, software visualization is visualization for software engineering; information visualization is the representation of large data sets and program or algorithm visualization is the structure and behavior representation of algorithms and data structures for pedagogical purposes.

2.1.1 Algorithm Visualization

Algorithm Visualization is considered to be the process of graphically illustrating the dynamic and abstract behavior or functionality of an algorithm and the internal state changes of its underlying data structures including interactive graphical displays of fundamental operations [33]. It uses computer-graphic related technologies to abstract the algorithms, data structures, their operations and semantics in order to produce a well-designed graphical representation of these abstractions [24]. There was rapid growth of this technology and related visual tools with the availability of high-performance graphics hardware, advancement in computing technology and the sustainability of software. Today, even the low cost PCs are competent to handle high-end computing visualization systems. Various concepts in computer science can be visually represented using visualization technology. The main objective of developing visualizations is to improve the understanding of the often non-trivial behavior of algorithms.

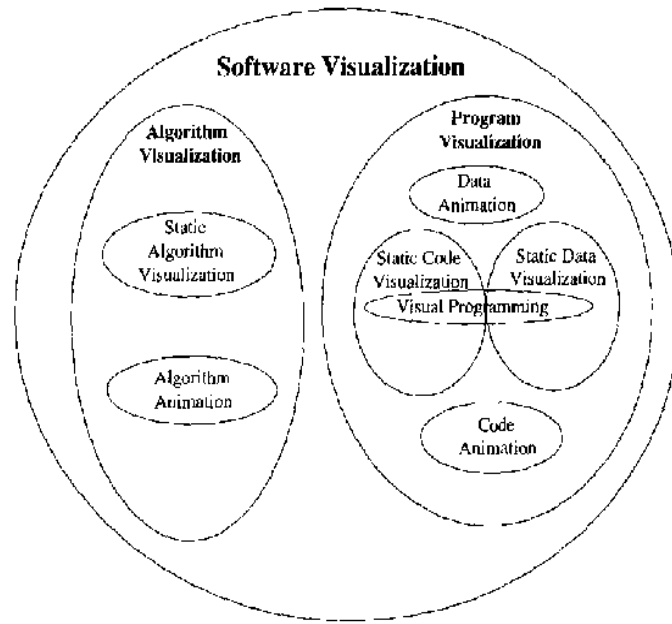


Figure 2.2: Visualizing Software Visualization [73]

2.1.2 Program Visualization

On the other side, Program Visualization is considered to be the dynamic graphical representation of the execution of programs or code [33]. It is often a mixture of data animation, code animation and visual programming. The main objective of program visualization is to improve student's understanding of the execution of program. Some of the techniques in program visualization include visualizing the call stack, code highlighting, supporting flexible execution control of the code visualization, and presenting information of the variables.

2.2 Algorithmic animation as a subfield of software visualization

To improve the algorithm visualization it is important to explore different forms of software visualization like program visualization and also to know the relationship among them is very essential. This helps in understanding the division of subject that is to be visualized, for

instance program visualization consists of programming language code for a particular implementation, where as in the algorithmic visualization the division will be more on the subject related to a high-level view of computation.

The division of program visualization is the combination of both static and animated code as well as data. For example, code can be viewed in an organized manner using Flow Charts [59]. Visual Programming (VP) is a way for the programmers to reach the users by offering computations using pictures that can be directly interacted with a computer [61, 62]. There are few tools of program visualization with features where both the code and data can be animated automatically. To observe the transformed variables, stepping through the code with a debugger is a common technique. Subsequent to the two levels of division of algorithm visualization static and animation, another important level is the static history. By seeing a pseudo code and the diagram of the data structure, an algorithm can be visualized; therefore by animating this pseudo code, the animation of an algorithm is feasible. The concept of static history is to produce a history of pictures of the operation of the algorithm.

CHAPTER 3

LITERATURE REVIEW

3.1 Design of algorithm visualizations

There are many algorithm visualizations being developed, but the designers of these visualization systems do not clearly specify their intention to support a particular task [Petre et al. 98]. The designing phase requires not only a creative mind, but also a very good working knowledge of any design framework for animation. According to Sami Khuri [32], visualizations can be used by all types of users in all kinds of tasks. But, there is no perfect algorithm visualization that universally satisfies all the requirements. A successful visualization should also consider the design issues like impressive representation and presentation, environmental factors, design and layout, color, graphics and user-interface, apart from the factors that make it more educationally effective.

For designing educationally effective algorithm visualizations, a significant investment of time, effort and resources is required. In fact, the process of designing visualizations needs to follow the software development life cycle that involves analysis of requirements, design, implementation, testing and maintenance. It is common to have these phases overlapping each other during the entire process. For designing algorithm visualizations, analysis of requirements is the most important phase as it decides on what and how to design. This key phase answers most of the questions like: Who will be using the visualization? How do you fit this in the curriculum? How to make it educationally effective? How can it be used as learning or teaching

tool? Is the system technically feasible? In order to design an effective visualization, it is important to analyze all these requirements clearly before starting to design and incorporate required features during the design phase. This avoids the wastage of extra time that is usually spent on the overlapping phases.

3.2 Analysis of requirements for algorithm visualization development from literature

Sami Khuri advises the visualization designers to consider the following different parts of the analysis of requirements for the development of algorithm visualizations [32]. These factors must be considered in order to design effective visualizations.

3.2.1 Users' Analysis

While developing algorithm visualizations, designers must know who the actual users are. Understanding these users determines the system's content, breadth, depth, organization and information presentation. These users can be divided into four roles which include students, educators or faculty, researchers or developers. More than one of these roles can be assumed by one individual.

These users can be further categorized into novice and experienced users. A system is supposed to be developed keeping in mind both the novice and experienced users, but ideally that would not be the case as it is difficult to develop a system for both the kinds of users. A single system cannot satisfy different levels of expertise. Beginners find difficulty in learning the system by trial and error method as they will have problems in mapping the real world entity into a programming object, thus forming misconceptions. Having a clean design with relatively few features is really essential for the beginning level users.

Expert users on the other hand, will like to play with the code to see how it works with various experiments and will then try to modify it as required. For example, they would want to see how the program works after modification, how the code really works when integrate these modules with other tools. It is also interesting for them to understand the behavior of algorithms which sometimes require their ability to switch between algorithm and program views.

3.2.2 Needs Analysis

The needs of the user group who use the algorithm visualizations must be addressed carefully. Different users or students have different learning capabilities, so it is very important to understand their needs individually. Some of the factors like the motivation, learner's expectation and locus of control also play a significant role in learning. In this context, locus of control refers to the extent to which the individuals can control the events on their own that affect them.

Some students who tend to explore things more and learn independently would prefer the "hands on" approach. While other students prefer to be led through the entire chapter by the instructor or the computer that controls the flow of the lesson. For these types of users, graphical demonstration of the effects and the animation are really useful. The algorithm visualizations must graphically exhibit the effects of the algorithm on the data structures. It would be very helpful for the students if they have animations shown directly with explanatory cues in various forms like short, on screen, textual notes and also if they were given the control of the speed at which the algorithm was animated.

The crucial part of the interaction design would be the ability to simultaneously view the algorithm execution path and the data structure state. The users who prefer an active interaction

would need some toolset that would be similar to that found in the program development environment. That is, “debugging” style software that allows breakpoint setting, single step execution and the monitoring of key variables is very essential for visualizing the algorithms.

But, before the implementation of the algorithm visualization system, the designers should ask the users if they want the information to be presented in the same way, i.e. by visualizing all the algorithms including the basic simple programs. Finding out this information is essential because the effort or the time should not be wasted in visualizing simple algorithms. If the amount of data is small, or if the data structure is simple, or if the relationship of the objects is important when compared to their movement, then perhaps static pictures would be sufficient. On the other hand, if the data is large, or if there are complex data structures, or if the movement of the objects is required where the relationship of the objects changes over time, animation would most likely be the correct choice for the presentation of the algorithms.

3.2.3 Task Analysis

While designing algorithm visualization system, designers have to mainly concentrate on the system’s intended goals and they have to select the required content accordingly. Different users have different situations in which they will be using the visualization system, such as the creation of new animations, interactions with existing visualizations to understand the algorithm’s behavior, or debugging the algorithms visually. Each of these situations would require a different kind of visualization system and it would be difficult to develop a different system for each situation.

Systems can be designed for classroom teaching which just shows the algorithm view and by hiding the code view so that the students’ attention can be kept away from the implementation

details. Students cannot do any kind of modifications to the settings or implementation data in the case of the systems that are developed for classroom teaching. The other type of algorithm visualization systems are the systems designed for user's exploration where they could change the settings, input various data sets, change the speed, move one step backwards, or move between algorithm-level display and the program-level display.

If the novice users use this system and find it difficult to make the animations and design their visualizations, then their time and effort is all wasted. So, the systems that are developed for this purpose must have a powerful editor that allows the students to map the graphical objects to the data structures automatically.

3.2.4 Information Analysis

Information needs to be analyzed in order to determine in order to determine how efficiently and effectively it could be visualized. Viewers or the users must concentrate more in figuring out what the picture or the visual presentation is instead of having a regular set of visual conventions such as denoting one item with one color, another item with another color. It would be more feasible if one spends more time on the interaction among visual images rather than trying to understand and follow the algorithms. The information which is graphically represented is completely dependent on the concept that is being visualized.

In order to understand various characteristics of the information, it is often helpful in providing the multiple views of the same system. These multiple views include the graphical views of changing program data and also simultaneously its corresponding view of the executable source code. For example, the figure 3.1 above gives the required information details about the Quad tree compression algorithm for bitmap images.

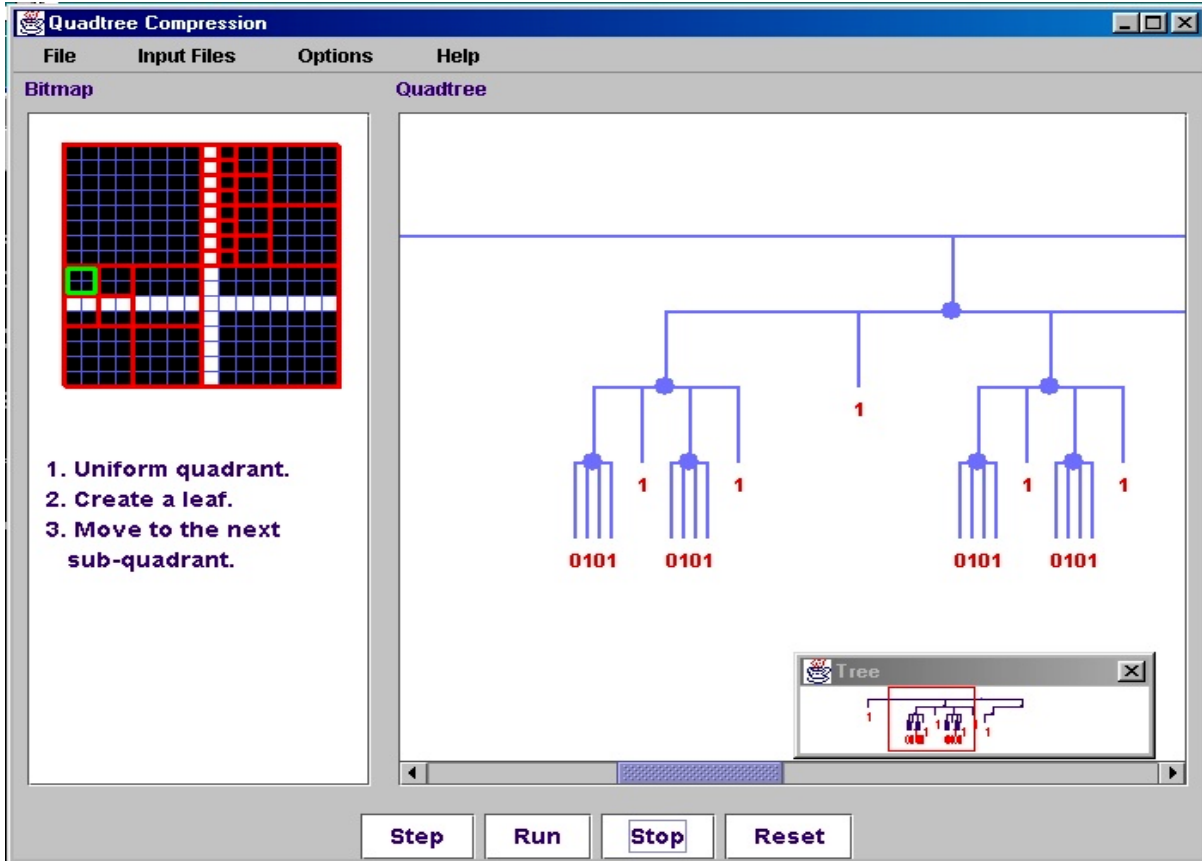


Figure 3.1: Graphical representation of information in Quadtree Compression Algorithm [32]

3.2.5 Scope Analysis

The scope of the visualization system can range from single-purpose visualizations (e.g., a visualization of a specific algorithm), to specialized systems where algorithms are mainly concentrated in a specific field of computer science (e.g., graph algorithms), and finally to the general purpose systems that are designed to provide visualizations for programs or algorithms in any domain. Single-purpose visualization systems exemplify one algorithm or a set of related algorithms in detail. The general purpose visualization systems ideally include the animation of any algorithm. It is tempting to believe that the greater the range of algorithms that are animated,

the more desirable the outcome is. This must be tempered with the reality that the increased flexibility causes an increased level of complexity in the system.

In general, if a system confines its animations to the algorithms in one field, it would be difficult to represent the algorithms in other fields. Designers of the algorithm visualization systems typically follow the following recommendations:

- Design small systems first and do not try to provide everything that is possible in the beginning. Provide whatever the designer can which has high quality, beneficial to the users and is visually attractive.
- Designers must have to plan a phased growth. In this rapidly evolving world, the visualization might change and grow over time. The development of these systems can be well planned for growths with the help of object oriented design and careful documentation of the programs.

Algorithm-specific visualization systems might not allow the addition of new features in the system. So, this addition of new features must be planned to add the new features over time and perform the upgrades whenever available.

3.2.6 Resources Analysis

Usually, designing a visualization system takes more time than expected. Designing these systems is not as simple as we think, especially when designing the appearance of the visualization. There is a great demand for visualizations in computer science and it demands many computer resources like the CPU speed, networking, RAM and hard disk memory sizes, monitor size, color display panel for output equipment and keyboard, audio and video input

devices. And if these visualization systems are over the internet, then it would require large space and high speed internet too as resources.

Selection of the specification method is also another important purpose of resource analysis in designing the new algorithm visualizations. Some of the specifications are the declaration, predefinition, annotation and manipulation. In the annotation method, the important steps of the algorithms are annotated with attractive events which implicitly call the graphical operations that finally execute the animations like the movement of items or rectangle, change of colors etc. When the program pointer reaches the important steps of the algorithm that are annotated during execution, the special events are created and then forwarded to the various views of the animation. These views correspond to the interesting events by respective animations. For example, the mapping between the algorithm/program's state and the final image can be specified arbitrarily or by declaration. The changes in the program's state will be reflected in the image immediately.

3.3 Academic experience with algorithmic animation

The majority of visualizations of algorithms available on-line are used by students for their course work and as classroom supplements. As an example of evaluating the impact of algorithmic animation in an educational setting, consider an evaluation conducted at Auburn University [69]. In this evaluation students were divided into two groups, an animation group and a control group. They were given the same set of algorithms to both the groups in a different manner, in animation group the algorithms were given in an animation written with the Toolbox system and for the control group the same algorithms were given in a textbook fashion with pictures. The performance results from the pre-test for both the groups were same and the post-

test results confirmed that the animation group was more proficient than the control group. These are the following observations made based on the post-test results.

- The students in the animation group had 74% correct answers in the test, compared with 43% in the control group.
- When two sorting algorithms were learned, the percentages of correct answers were 63% and 44% for the animation and the text group, respectively.
- Only when the control group had to solve additional homework questions did the proficiency of both groups become comparable.

In the above evaluation, in spite of getting good results for the animation group, the actual number of sorting algorithms used is just four and comparison was just between the two groups. To get the actual comprehensive results, there are lot of measures need to be included like teaching of complete courses in similar fashion that dividing the students into control and animation groups, all the different possible characteristics need to be included, and the quality of teaching in each group should be same. Consequently, there is a need to have wide-range of studies to evaluate the effectiveness of algorithmic animations over students. It is important that students need to make a mental connection between conceptual features of the algorithm with the algorithm itself, and there is always a chance that an intelligent student can recognize these connections compared to a normal one. Stasko and his colleagues [39] have studied and concluded that the above mentioned method of evaluating is not very helpful for the students in the educational field as expected. Another observation from [18] explains how the complexity of an algorithm can vary the results between the groups, for instance while learning a simple algorithm the results were much better for animation group than the control group, and in the case of complex algorithms the results between the two groups were similar.

The extant research literature suggests that an animation is more understandable for a student if it is involved with other communication system like an audio explanation. This can be easily implemented in a class room where an instructor can act as another communication system with clear explanation of each step, while playing the animation.

Another research project [70] observed a visualization-based learning process in a classroom using an entirely different approach. This project required students to write their own algorithm visualizations as a part of assignments. Through such assignments students are made to visualize the algorithm with suitable models, granularities, and steps which, to the student, demonstrate the algorithm in a best possible way. The data suggested that students benefit from these kinds of constructions in visualizing an algorithm. Furthermore, studies from Hubscher-Younger [71] have supported the notion that the advancement of understanding algorithms can be better if the students are able to describe and script their own visualizations. Another research project called “active algorithm learning” [72] reported on a system that presents animations and asks questions in such a manner that the nature of the animation is based on the user’s answers.

Two underlying theories that play a major role in learning algorithms through animations are epistemic fidelity [54] and the dual coding hypothesis [54]. In epistemic fidelity the visualization emphasizes the mental models of the physical and logical world, under the assumption that humans intrinsically have these mental and relational models. In the dual code hypothesis, data can be coded in either verbal or non-verbal way that would make the knowledge transfer efficient from teacher to student [29].

CHAPTER 4

BEST PRACTICES FOR EFFECTIVE VISUALIZATIONS

In this chapter, the best practices or the key requirements that an effective visualization system should possess are discussed in detail. After a thorough literature review, a set of pedagogically essential features are captured in nine best practices.

In the earlier stages of development in the field, the main focus of the research of algorithm visualization is on the design and development of the algorithm visualization systems or the tools. Now, however, the primary focus of the research is oriented more towards how to design effective visualizations, how to effectively use existing visualizations, how to engage students efficiently with the systems and how can this technology be best adopted into computing education [29, 33, 34, 35, 36, 37].

Educators are constantly exploring new ways of engaging students with algorithm animations and finding solutions to hold the learners' interest and attention with the visualization, and to improve instruction. Though this process requires time and effort, according to [11, 33], most of the teachers find it satisfactory and convenient for explaining algorithms through animations, when compared to traditional teaching. Algorithm animations have been helping not only the teachers in illustrating algorithm operations effectively, but also a vast number of students in learning the difficult parts of algorithms and data structures. Some studies reported significant improvement in learning by using visualizations; whereas others indicated no difference in the improvement of students' knowledge after using visualizations when

compared to traditional learning. Hundhausen et al. have shown in their meta-study of algorithm visualization effectiveness [29] that the students' interaction and involvement in the visualization is more important than just viewing it.

In 2002, the ACM conference ITiCSE sponsored a working group that proposed a research framework [11], with lot of open questions for researchers in the field including interaction taxonomy, and number of testing strategies for the evaluations of visualizations. It has proposed six levels of student engagement in learning: no viewing, viewing, responding, changing, constructing and presenting. Much research has been carried out within this area over past few years.

The following sections discuss one of the nine best practices distilled from the literature.

4.1 Need to support flexible execution control of the algorithm visualization

In order to provide smooth execution while playing the visualization, visualization systems should support a few mandatory execution controls on the user interface that allows the users to interact with the visualization by controlling animation steps, start/stop, speed, etc. Whenever the user wants to review any random step or to pause the animation for detailed understanding, these controls help in performing user specific actions. The design of the user interface is highly important for a good animation. It has to be designed in a way that is not confusing or clumsy to the user. All the controls and required data should be presented in an organized and appealing way so that it not only enhances user interaction but also has attractive look. The graphical user interface consists of simple widgets such as input fields and controls that enable the user to control the flow of the animation. These controls should be flexible which includes the ability to move through the visualization both forwards and backwards (for example,

[38, 39]). The visualizations can also be run automatically based on adjustable speed. The execution controls on an effective visualization are more similar to the ones on a video player, which has controls for the following actions: play, stop, pause, single step forward and single step backward, continuous advance, advance to the end at once and backtrack to the beginning at once. A similar user interface is shown in figure 4.1 below from the ANIMAL visualization tool.



Figure 4.1: ANIMAL's animation display control tool bar [13]

Different users have different levels of interpreting, grasping and understanding the content provided in visualizations. Even a single user may want to use the visualization at different rate or speed while working at different stages of it depending on his understanding capability. All the users may not adapt to it or grasp the information the first time. So, an effective visualization may also have to have control to vary the speed of presentation, and the learners should be able to run their test cases again and again if needed. Below is the figure that shows the speed control tool bar from the ANIMAL system.

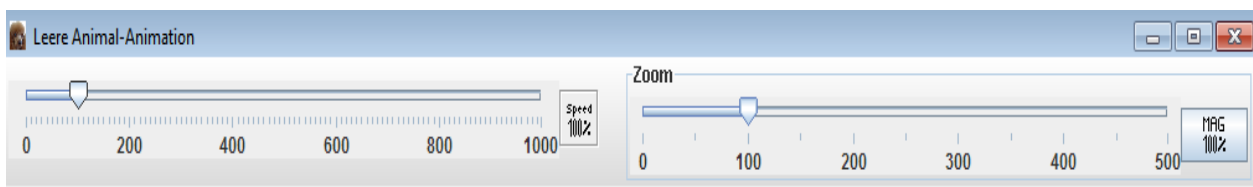


Figure 4.2: ANIMAL's speed control tool bar [49]

Visualizations are more effective and efficient when the user has enough options to steer through the steps in appropriate directions. While designing algorithm visualizations, the designers should make sure to include interactive controls for the user.

4.2 Visualizations prompt users for answers to questions, predictions, etc.

To enhance the student's learning, this practice is one of the best approaches in the design of visualizations. The support of dynamic stop-and-think questions allows the users to check his/her understanding of the execution, description and behavior of an algorithm [11, 40, 29, 43]. The use of probes or pop-up questions also promotes high interaction of the user with the visualizations that greatly stimulates thinking and fosters self-explanations [22]. A built-in editor can allow the developers to create multiple choice questions for the learner and to set them to be triggered periodically at specific stages. At the same time, it can help the learners to self-evaluate their performance and knowledge of the algorithm. Designers are thinking of ways to engage learners and retain their interest and enthusiasm throughout the visualization. Visualizations are more effective when there is high interaction between the animation and the users through non-trivial questions that force them to answer content related to the algorithm and predict the future behavior. There can be two kinds of questions for the learners [11]. The first type of it is random questions that pop-up any time at an appropriate context. This type focuses on improving learner's understanding on specific issues, challenge their understanding and promotes self-evaluation on how they perform. The second type of question is given at some critical points during the visualization execution where the learners cannot proceed further until they answer these questions right.

Figure 4.3 shows a screenshot of dynamic question pop-up from JHAVE for a Binary Search Tree. This is a first kind of question where it allows the user to self-evaluate in order to enhance learning and encourage student participation. Other systems like HalVis also pose similar interactive pop-up questions to the user periodically during the execution of the visualization.

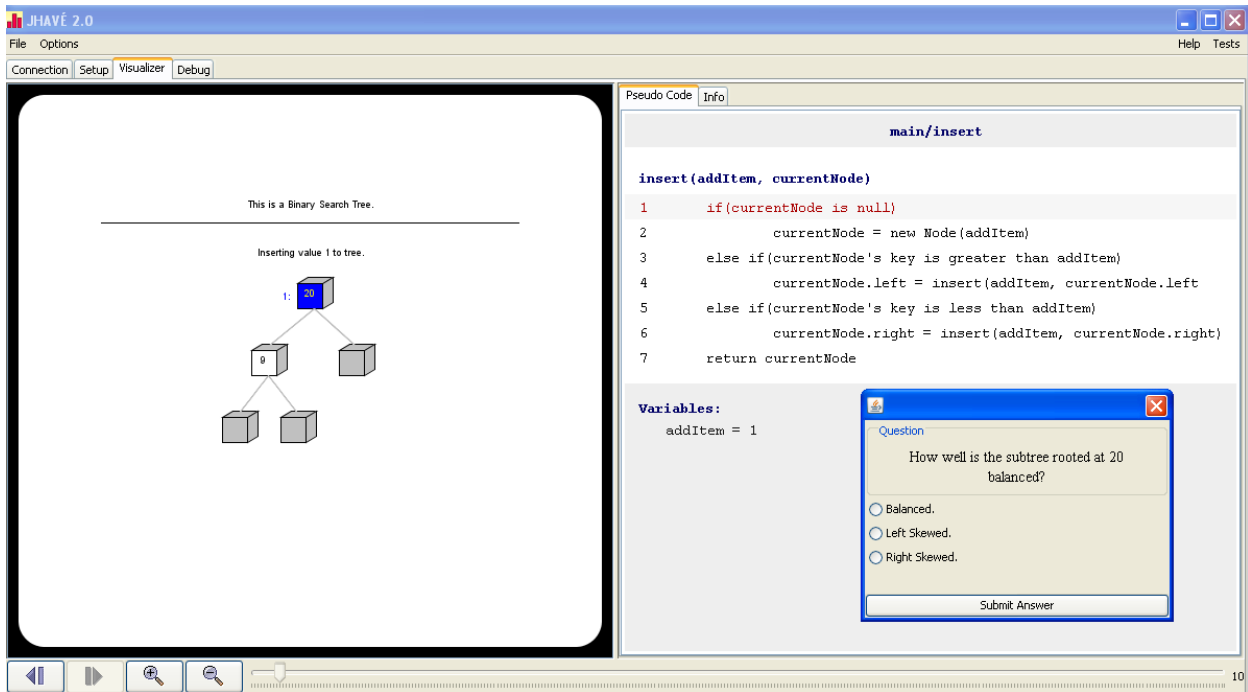


Figure 4.3: Representation of dynamic questions during algorithm execution from JHAVE [50]

If the visualizations support prompt and dynamic feedback [11, 42], they can have more positive effect on student's learning. For example, when the questions are answered wrong, immediate feedback about their mistake gives good understanding on their current knowledge. Other kinds of feedback systems are well illustrated in [41] by Korhonen and Malmi. The visualization system they presented has graphical representations of algorithms and during

execution; the learners are required to manipulate the visual representation for algorithm simulation. The visualization is designed to produce automatic and prompt feedback on the learner's performance explaining the correctness of the simulations. Sami Khuri says that the user should be forced to interact with the visualization at least every 45 seconds during the execution. This kind of interaction helps the learners understand every step of the algorithm or data structure better before proceeding further.

4.3 Provide a context for users to interpret the visualization

Though the basic purpose of visualizations is to provide a friendly user-interface that assists learners in understanding concepts behind it, sometimes visualizations are difficult to interpret. Designers need to plan well in order to provide a good interface that helps the users to interpret it at first glance without any ambiguity for it to be an effective visualization. The common difficulty for the learners using visualizations is mapping that visualization to the underlying data structure or algorithm it is developed for.

This mapping content and the underlying meaning of the animated representations can be clearly explained to the learners in two different ways [11]. One of it is to embed text description or narration into the visualization against each step during the execution. It can dynamically describe the relationship between the visualization and the concept behind it for every step so that the user understands the flow and algorithm clearly and will be able to predict the next algorithm steps. Before the visualizations starts, it can also briefly describe the theoretical concept and purpose of the algorithm that helps the learner to interpret the visualization in right sense.

Figure 4.4 below shows a graphical representation of the longest common sequence algorithm using ALViE (Algorithm and data structure Visualization Environment) tool. Below the visualization is a separate window for the messages that clearly describes the operation of algorithm in every step in the form of text. The corresponding source code description for the text described is also highlighted in order to match both of them for better understanding.

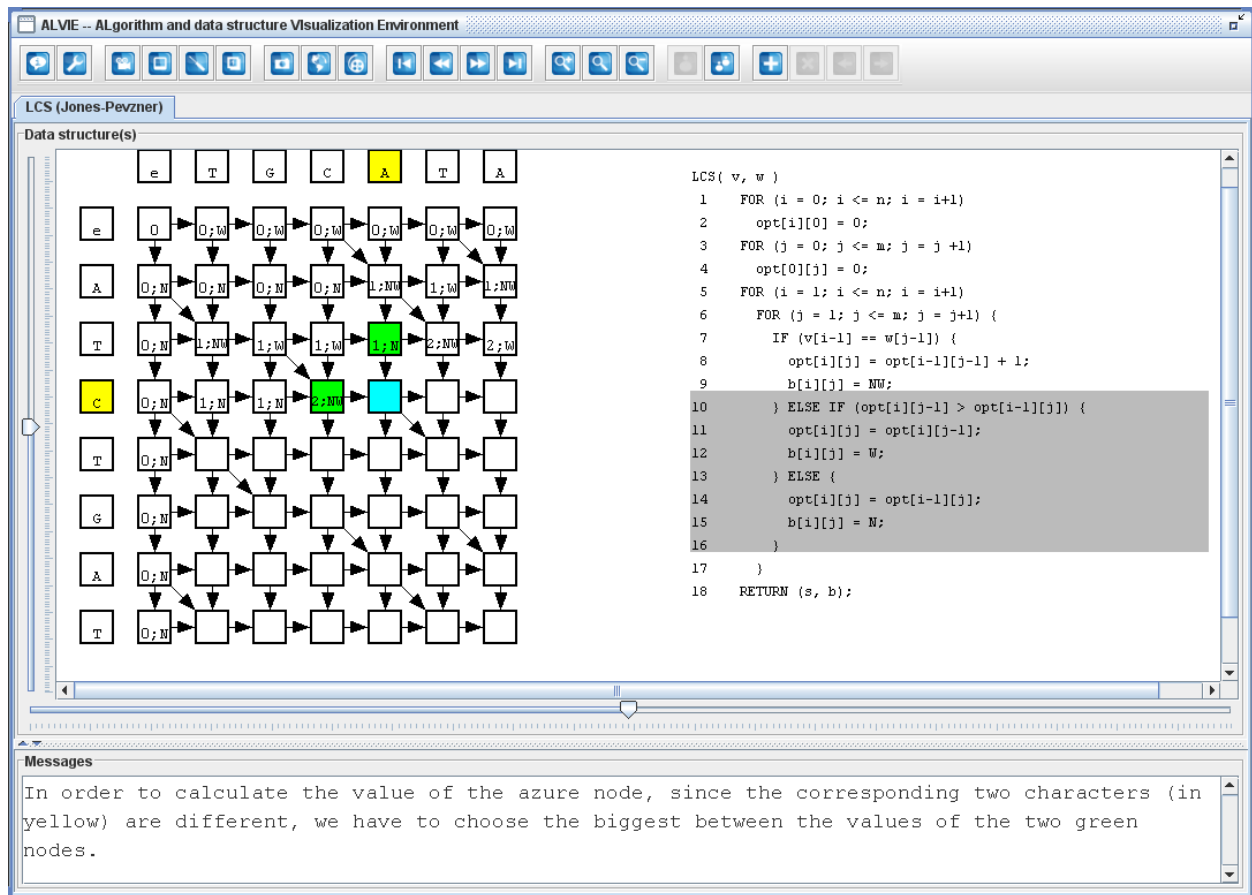


Figure 4.4: Representation of textual description for easy interpretation of the visualization

The second way to enable easy interpretation and understanding of the visualization is to allocate sufficient time for explaining the general and basic concepts of a particular algorithm or data structure during the course instruction. This can be achieved in different ways like having

text description in the corresponding visualization website link or in lectures, labs; just before the learners open/click the visualization needed.

All the algorithm visualizations operate on different state changes of the data structures used. For better interpretation, algorithm visualizations need to be presented in discrete segments with each state or state changes shown along with the explanations of particular operations.

4.4 Provide multiple views or representations

Often it is beneficial to provide different views of an algorithm simultaneously [11, 22, 42, 74], like the animation of algorithm, pseudo-code or source code view, textual description view, physical implementation of algorithm and logical view. These views can be shown on the visualization simultaneously with appropriate synchronization for clear and better understanding. The user can also be given a choice to page the windows to avoid complexity so that only one or more windows can be seen at a time. But the designer should make sure that all these windows are properly visible to the user when he starts the visualization and that the actions in one window are appropriately related to the corresponding actions in other windows. All the windows should be coordinated well to display consistent information. Multiple views of an algorithm to the user can facilitate in-depth and better understanding of the algorithm logic and operation. This practice highly adds advantages in the educational perspective.

For algorithm visualizations to be effective, they need to have a program animation view, that has either pseudo-code or source code with code highlighted as the animation executes [11]. Or they can have program animation view showing just part of the code that is getting executed right then and this view keeps changing for every step. A mandatory view is the algorithm or data structure animation view that can be accompanied by textual explanations of the steps, that

provides bridge between the algorithm animation view and the program animation view. Visualizations can also have views that show the input data sets allowing users to input values and display the output in another view either graphically or in text depending on the algorithm and its visualization. Many of the visualizations have a physical view and a logical view for those using data structures such as heap. Heap can be shown as an array in physical view and as a tree in logical representation. These different views need to clearly show the inter-relations and connections amongst each other. An example of this can be seen in the figure below that represents multiple views of Huffman Coding algorithm visualization. Other systems like HalVis are also designed similarly to show multiple views.

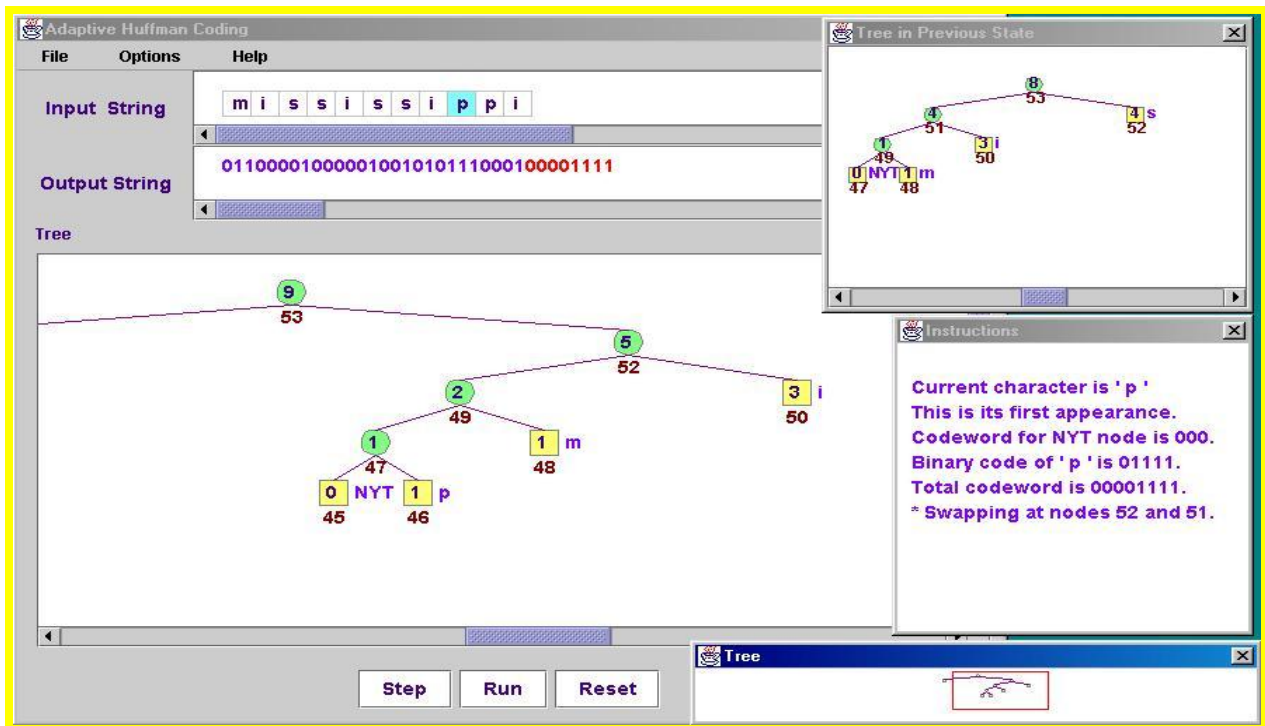


Figure 4.5: Representing multiple views [23]

It is important to make sure that each of these views must be easy to comprehend when viewed separately and each view needs to show only few aspects of the algorithm [23]. This approach helps the learner by avoiding the need to remember the algorithm states from previous steps. Also the designers should carefully synchronize the display by clearly distinguishing each view's matching content. This can be attained by choosing same color, size and shape (circle, square etc.) for the related data in different views.

4.5 Allow user-specified data sets

It is assumed that the concept of active learning is superior and more effective than passive learning [29]. In order to actively engage learners in the visualization, designers have to make sure to include the option where the users can construct their own sets of input data (for example [2, 41]). This practice allows the user to deeply explore the algorithm behavior and discover how it behaves for different input sets. The user is free to test the algorithm for its best and worst performances and roughly determine its complexity based on results of the range of input data. By allowing the learners to use their own test cases on the algorithm, he or she can get important questions on the algorithm's behavior and execution clarified. This may ultimately improve the self-efficacy of the student and encourage further learning. The figure below is an example of allowing user specific input data set in Quicksort algorithm from AIA (Algorithms In Action) system.

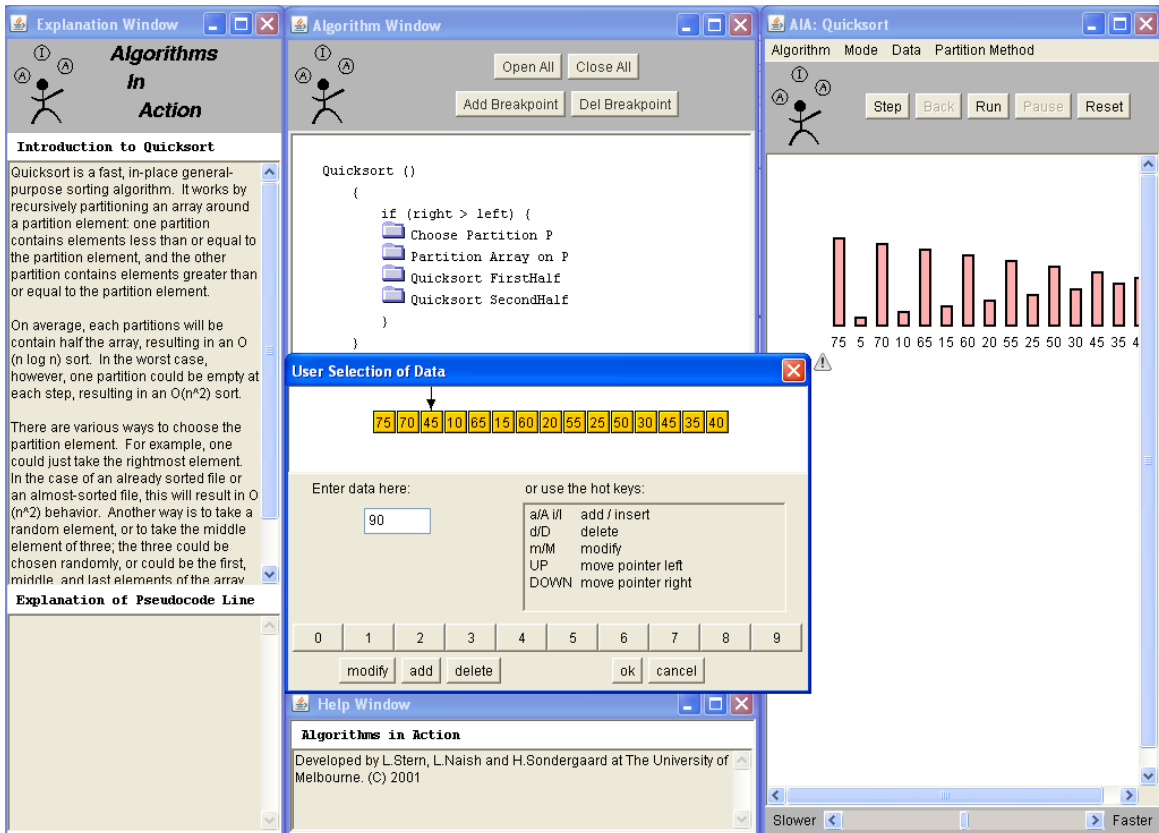


Figure 4.6: Allowing the user to input any data set to explore algorithm behavior

During the design phase, designers of the visualizations should consider the validation of the user specified input data. The input tool for inputting self-constructed data set needs to validate for errors and invalid input and report it back to the user. Also these input tools should be designed as simple as possible in order to avoid interface complications. This feature might also augment the understanding of step-by-step or procedural behavior of any given algorithm. A given test case might help the learners trace the algorithm's variables and data structures in detail for a valid input data set. A high level of conceptual knowledge is indeed required to understand the procedural behavior of an algorithm.

4.6 Provide underlying source code and program execution

Software visualization includes both algorithm and program visualization as discussed earlier. In order to strengthen student's learning, program execution view or presenting source code or pseudo code is equally important while viewing the visualization [42, 74]. This is similar to providing a context for the user to interpret the visualization better, as discussed in section 4.3. Typical program visualization view includes code-highlighting, showing program variables information and call-stack [40]. The main objective of presenting the environment with program execution features and underlying source code is to understand the execution of programs line-by-line and simultaneously explore the state changes in data structures used by the programs. This approach is highly accepted as it not only has the above important features but also helps increasing knowledge of the students even with no prior programming experience. This indicates that providing underlying source code and program execution enhances novice students' learning too [40]. The figure below shows the source code behind Huffman Coding using the Auckland visualization system and highlighting the statement that is currently executed along with the textual description, which allows the user to interpret the visualization accurately.

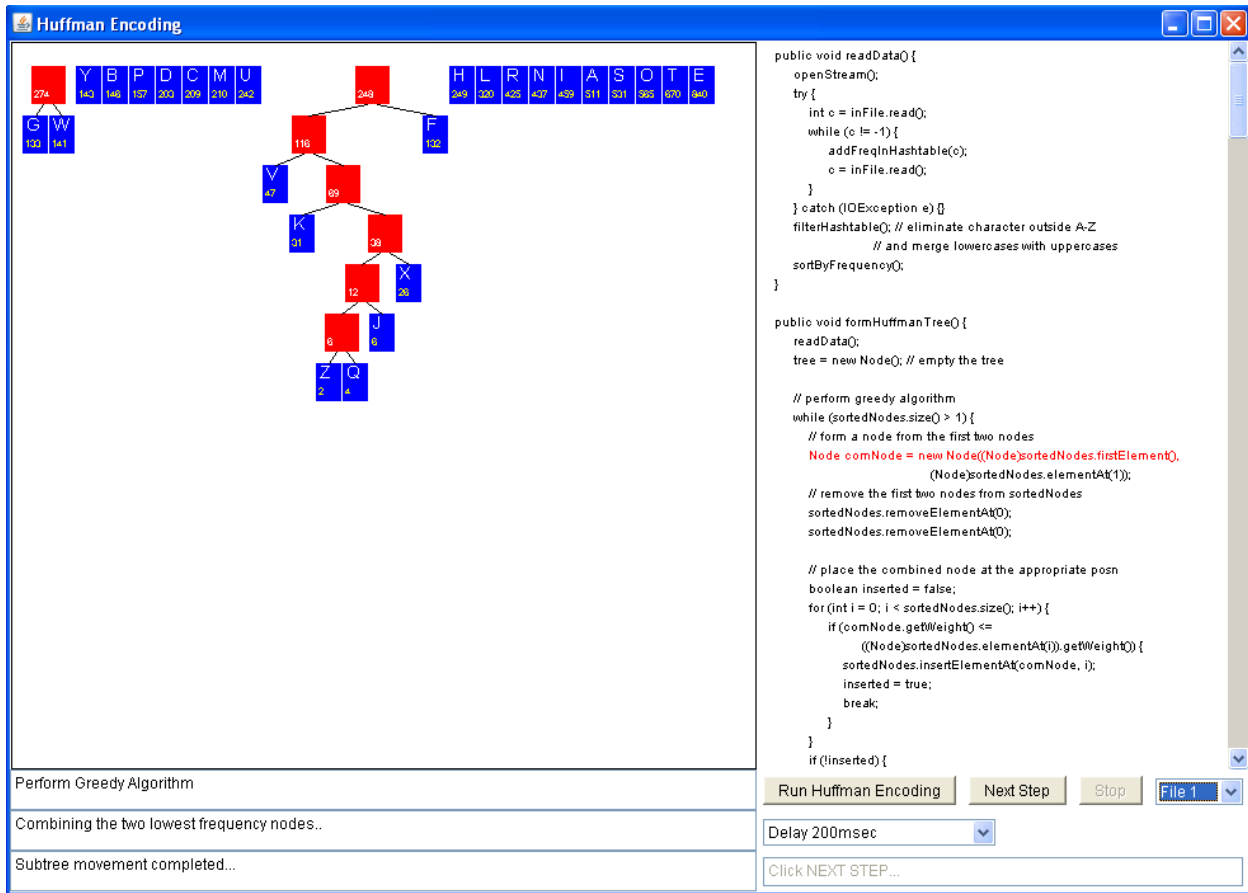


Figure 4.7: Representation of program visualization and code highlighting

Tracing the execution also helps the user learn better, but very few visualization systems show the call-stack information and state changes of data structures during visualization. jGRASP [95, available at <http://jgrasp.org>] is one of the few exceptions, as shown in figure 4.8 below. This information helps the learner keep track of variables and data and can predict future steps and state changes. The call stack is the view of what is executed in the program previously showing the function calls and the returns between different methods. The call stack is also useful for understanding recursive programs. It is important to keep all these program visualization techniques synchronized and co-ordinated with the algorithm visualization.

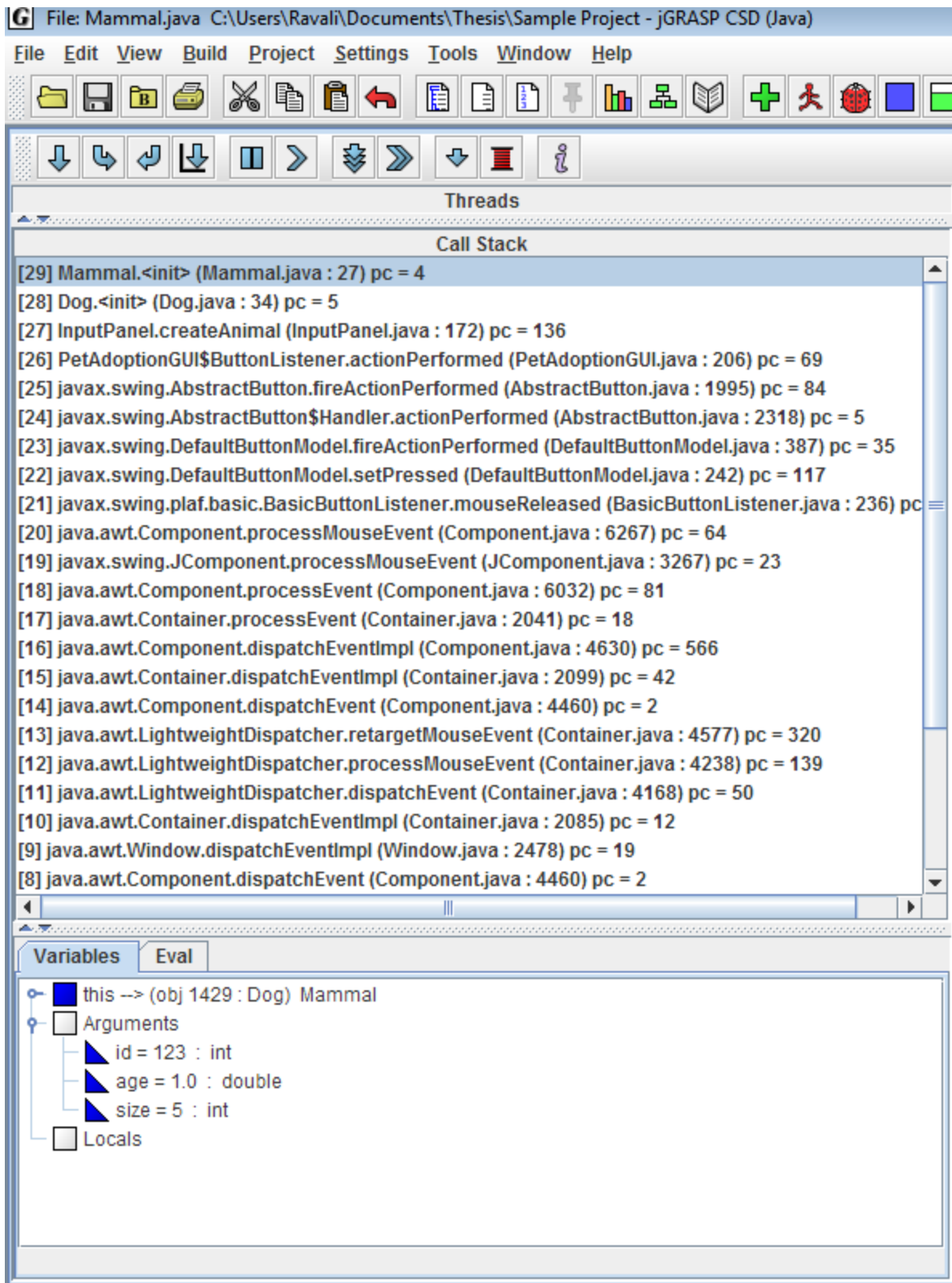


Figure 4.8: Representation of the view of call stack during a program execution

4.7 Can it be used as a lecture aid?

One of the main purposes of developing algorithm and data structure visualizations is to enhance student's learning either by teaching them in lectures or by self-learning. Most of the designers start developing visualizations with a notion of serving the students in their learning and design with a purpose of using them as a lecture aiding tool or a self-learning tool.

Animations are the software media that depict the execution of a program dynamically and are designed to assist learners improve their understanding of algorithms and teachers in facilitating learning. Preliminary results have shown encouraging and supporting results on the effectiveness of visualization systems on students' learning. Nevertheless, application of it in computing education is not that widespread. This disappointing result may depend on the usage of visualization systems as pedagogical tools by the teachers [45].

The results from literature propose that the use of such tools by teachers can be increased either by integrating other learning materials with these tools or by specifying the importance of the use of visualization of software to the students. As teachers are the primary connecting link between pedagogical tools such as algorithm and data structure visualization systems and the students, it is believed that the teachers need to have more important role and innovation in adapting them as teaching aid for students. Though the basic idea of developing visualizations is to augment student's learning, there are mixed results based on the experimental studies of pedagogical effects of algorithm visualization. Few studies show that visualization technology has positive effect as pedagogical tools on student's learning [21, 18, 16], while other studies do not encourage visualizations to be used for learning as their results had no effect on students' ability to learn algorithms [17, 19, 20]. Kann, Lindeman and Heller [46] propose an effective

way of imbibing algorithm animation into teaching as to have the coders implement the algorithm or any program as part of the overall learning experience.

The influence of visualization on education depends primarily on how well it is used and how widely the instructors use it for pedagogical purposes. Though instructors are constructive and innovative in trying to use visualizations in lectures, they are not highly encouraged due to the following five impediments that are based on a survey of SIGCSE members done by 2002 working group [11]:

- 93%: time needed for searching effective examples
- 90%: time taken to learn and familiarize with the new visualization tools
- 90%: time required to design and develop new visualizations
- 83%: lack of effective visualization tools for development
- 79%: time needed to adapt or integrate the developed visualizations to respective course content or to the teaching field.

Based on these results, it is clear why an instructor is not very likely to adopt visualization technology in teaching. The factors like amount of time and effort needed and the unavailability of efficient resources badly influenced the use of visualizations in teaching environment. It is discouraging to the teacher for using the visualizations when a significant amount of time is needed for learning the tool and then developing demonstrations and interactive lab exercises for the students. To overcome this, Naps et al. [14] have proposed an idea to provide high quality support manuals for the instructors. The availability and use of these materials may increase the satisfaction level of the instructors and ultimately increases the use of visualizations in teaching. After the usage, the instructor may highly recommend visualizations

into teaching seeing students' improvement in learning and performance. And the above mentioned impediments may also be minimized in the future.

4.8 Can it be used for self-study?

Many algorithm visualizations are designed so as to allow students to learn the material themselves by viewing and interacting with it. For the visualizations to be effective, designers have to spend time developing very friendly user interface that should not need any extra assistance to get acquainted. Students do not like to put much effort learning how to use algorithm visualization when their main focus is on learning algorithm behavior.

Visualizations interest the students more when they are actively engaged with the activities like allowing students to provide input data sets to explore the behavior of algorithms, providing dynamic stop-and-think questions about the visualizations, providing opportunity to make predictions about future algorithm behavior or state changes, and having chance to develop their own animations as one learns by practically working on it than just by viewing it [43]. Such kind of interactive exercises during the learning process not only keeps the students' interest in the visualization but also helps them to improve their knowledge significantly.

As the students try to learn about the algorithm by their own, it would be highly beneficial to the students if the visualization designers incorporate some kind of textual or audio descriptions and explanations about the algorithm and the basic theory behind it prior to the visualization view. Algorithm visualization can just show the execution of a given algorithm but the students should know why that is happening too, for which they need prior knowledge about the algorithm.

4.9 Can it be used as a debugging aid?

Visualizations can also be used as debugging tools where the users can verify whether their implementations of structures are working correctly. Apart from serving as learning systems, visualizations are also used for debugging programs and research in the analysis of algorithms. In the debugging context, data structure and algorithm visualizations and visual debuggers operate in a similar fashion in that they visually show the information about data paths and contents in the memory step-by-step [47]. Debugging requires overall understanding of the algorithm and its specific components. Through debugging, the software developers can get deep knowledge of the structures of classes and packages of object-oriented software and also details of the state of the program. Visualizations help the users debug their programs in terms of logical debugging and performance debugging.

Debugging is an important phase to have for those visualizations that allow learners to create their own visualizations. This approach helps the student designers to easily know the implementation flaws and can be fixed soon and simultaneously learn a lot through the mistakes as their incorrect approach visually shows them what happens if wrongly coded. The algorithm visualization clearly illustrates the steps that the user's code should follow or depicts the expected results of student's code visually. Having theoretical knowledge of the concepts, the student can then debug the output accordingly comparing it to the visualization. Algorithm visualizations focus more on the implementation; but their output is still conceptual and machine-oriented.

CHAPTER 5

DIFFERENT VISUALIZATION SYSTEMS

Algorithm visualization is a sub-part of software visualization that dynamically visualizes high level abstractions of the software. Many algorithm visualization systems have been developed over the past three decades. In computing education, students usually find it difficult to understand the dynamic behavior of data structures and algorithms that is often tough to explain in classrooms using blackboard. Many visualization systems have been developed to explain such concepts and make these easy to understand. A few significant visualization systems are described in detail in this section with their advanced features. Of course there are many other visualization systems and similar description could be given for them as well.

5.1 ANIMAL

ANIMAL is an interesting tool for developing algorithm animations that can also be used in lectures for teaching students and enhance their learning. Some of the visualizations developed [48] display only the animation without any active engagement of the user. Some of the visualizations require the knowledge of creating animations using API calls, because of which some computer laypersons may not be comfortable developing them. To avoid this problem, current visualization systems have taken the approach of using scripting languages to create visualizations. But even this may be challenging to the non-programmers to explicitly generate the scripts. The development of the ANIMAL tool erased these shortcomings of other

visualization tools. It is user-friendly to use ANIMAL as it offers visual editing during development of the animation. It has simple scripting language called ANIMALSCRIPT that is provided with animation generation API.

ANIMAL is more famous for having a set of powerful features that can be used as different mixtures for creating and displaying animations of data structures, algorithms and many other computer science related topics. By using various other visualizations, the users have always been showing interest in having textual descriptions accompanying the animation. ANIMAL satisfies this requirement by providing both source or pseudo code and textual comments with the animations. The main focus of this ANIMAL tool is to provide an easy learning and animation development tool to the users with wide acceptance of it in teaching and learning.

The full feature set of ANIMAL and its acceptance by a large audience positions it as a significant animation tool. It is a platform independent installation that runs in both Windows and Linux/Unix platforms. It is freely available to teachers and students and very easy to use as it does not need any programming skills to display or generate animations. More importantly it doesn't require network access for using it as students and teachers may not always have access to the network in labs, classrooms or at home. As mentioned earlier, this tool supports the provision of source or pseudo code along with textual descriptions. For easy understandability, it also includes the support for code highlighting and clearly shows the execution of the program. It also provides wide applicability to create animations in various other topics apart from algorithms and data structures. The figure below is an example animation of Linked Lists using ANIMAL.

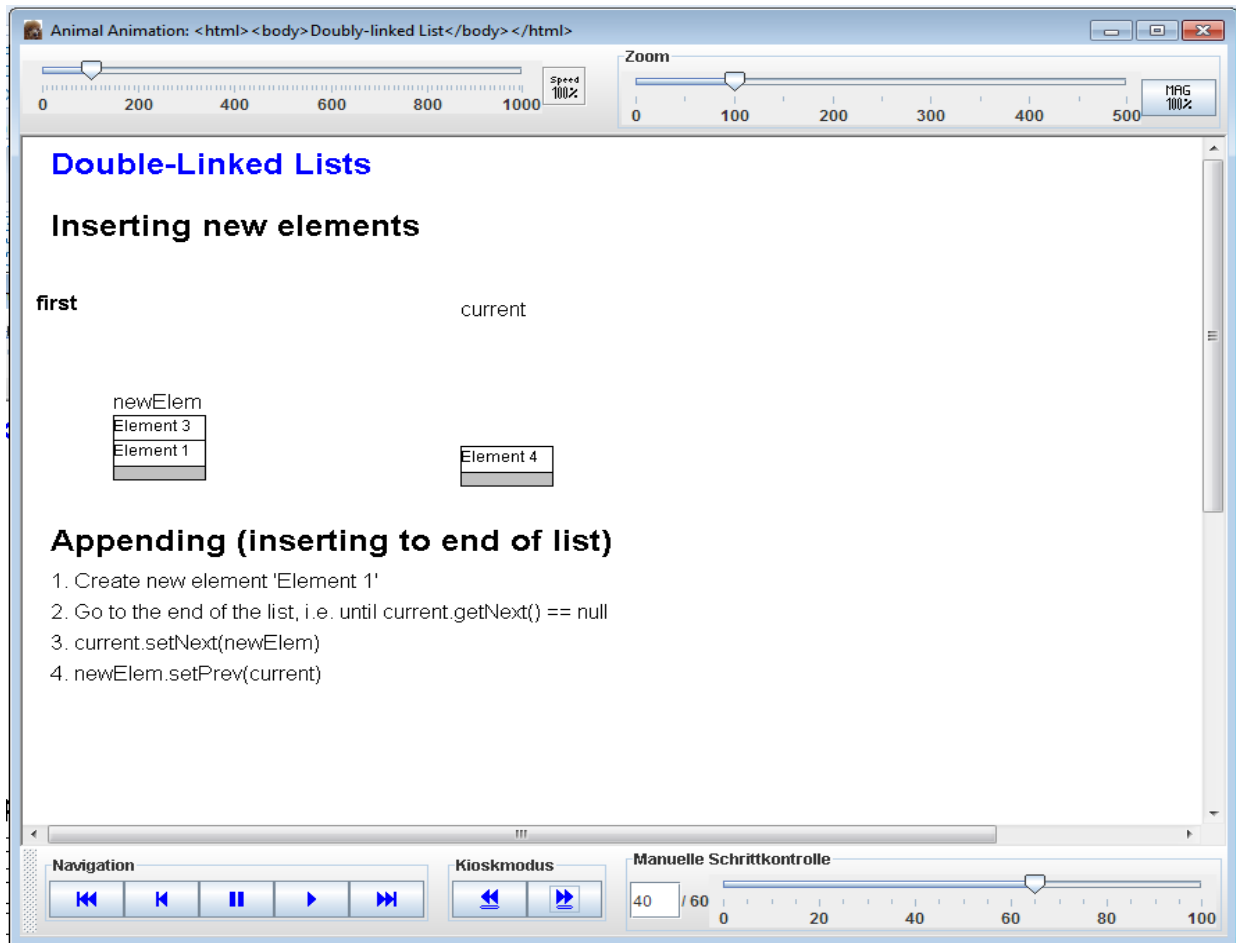


Figure 5.1: Algorithm animation using ANIMAL [49]

ANIMAL stands for “A New Interactive Modeler for Animations in Lectures”, which is written using Java’s Swing library. Animations are drawn and edited on a drawing pane for it to be easily usable. Thus the animation designers can create good animations even without knowledge in programming. More specific objects are created from a generalized set of given objects and they can also be reused for further animations. It takes much less time for a novice developer to familiarize themselves with the tool and the drawing interface. Developers, depending on their level of programming expertise, may choose to use either the scripting interface or the ANIMAL API that automatically generates required files into the ANIMAL’s

built-in scripting language. Both the approaches are the same in functionality. The animations have video-player like user controls and can also be scaled up or down. The added advantage to this tool is that the storage format is simple and easy to read. Based on the evaluation in [49], most of the students want to continue the use of animations in teaching.

5.2 JHAVE

JHAVE was developed to not only be graphically impressive but to also be an effective pedagogical tool. The figure below shows the execution of Binary Search Tree from JHAVE. Studies based on engagement taxonomy from a working group report on algorithm visualization effectiveness [11], have helped in designing effective activities in the visualization systems for more active engagement of the students.

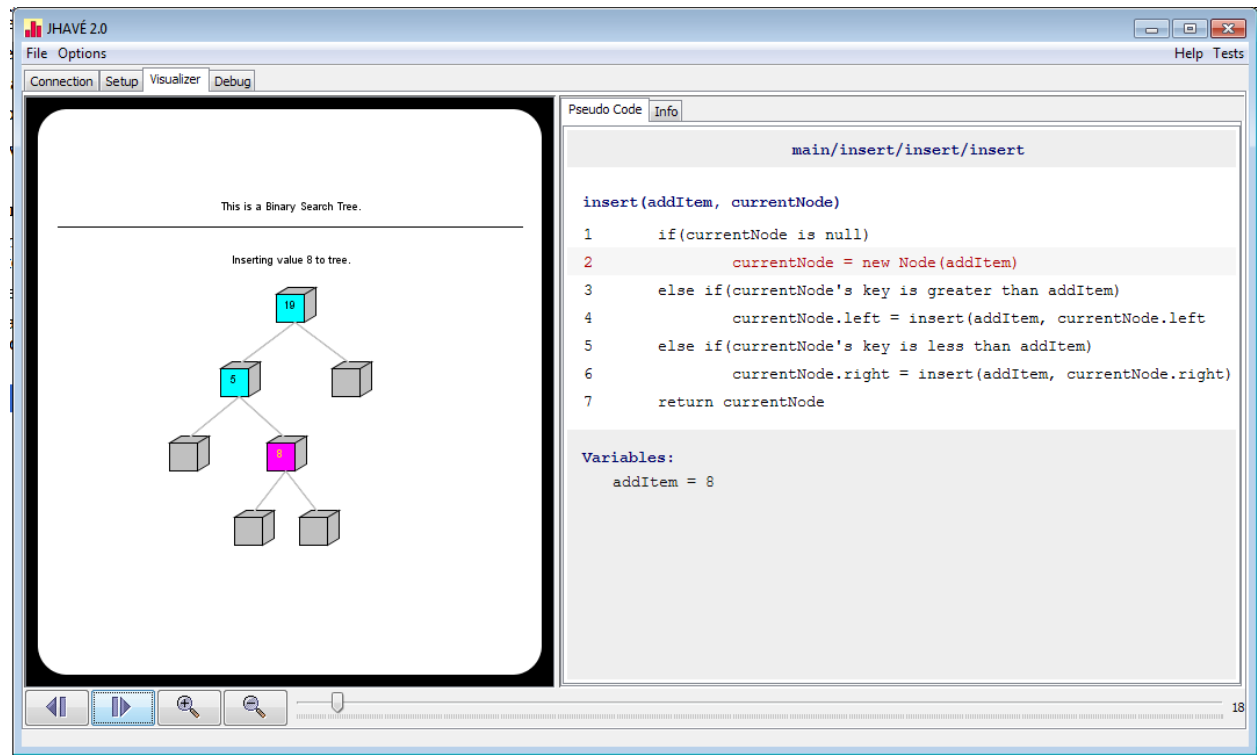


Figure 5.2: Representation of algorithm execution from JHAVE [50]

Graphics or the animations alone do not make the student understand the visualization completely. To make visualizations more effective, it needs engagement hooks that actively involve students in the visualization activities. But this is achievable at the cost of more effort during development. To overcome this, a new tool JHAVE (Java-Hosted Algorithm Visualization Environment) was developed. JHAVE is not only an algorithm visualization system, but it also provides a support environment for many algorithm visualization systems, called *AV engines* by JHAVE. The system is provided with lot of interesting features (described below) that synchronize well with the student's understanding. The interface consists of a standard control set like that of VCR's with navigation controls to allow students to step through the execution of the algorithm visually. Hence the GUI is not dependent on the AV engine that is used for the graphics.

The tool also provides information and source/pseudo code windows. These are the HTML windows controlled by the visualization designers to display either the static or dynamically generated significant explanation on what the student is seeing in specific. The information pane briefly explains the high level theoretical information while the pseudo code window displays the respective pseudo code of the algorithm with code-highlighting as the algorithm is executed in the animation. The students or the users are also allowed to input their own data set to explore the algorithm behavior and test against their anticipated output using the visual display. The designer also has the option to pop-up random and dynamic stop-and-think questions (as shown in figure 5.3 below) in terms of fill-in-the-blanks, true/false, multiple choice questions and can also make them mandatory so that student can proceed further only if he answers them right. This would facilitate them to make predictions about future steps or state

changes. JHAVE also provides a set of class libraries to help the designers develop their visualizations.

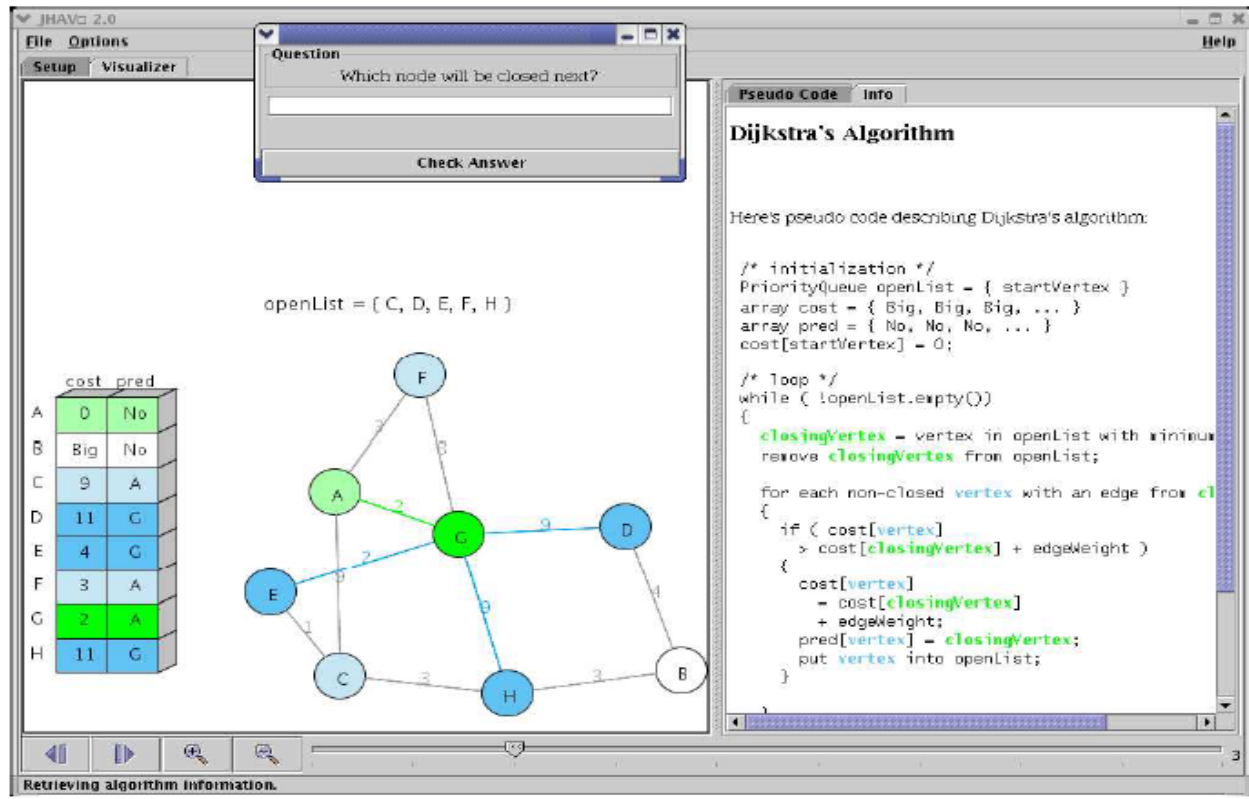


Figure 5.3: JHAVE representing algorithm visualization with pop-up questions [50]

As JHAVE has the client-server architecture, the script-producing programs are controlled by a central server. Hence, the developers are free to develop visualizations using any programming language and can be viewed in JHAVE environment.

5.3 jGRASP

In order to effectively use the visualizations during the development of code, multiple views are to be automatically generated in a synchronized way without leaving the Integrated

Development Environment (IDE). The jGRASP IDE, available at <http://jgrasp.org>, is developed to provide effective, dynamic and state-based visualizations of various objects and variables created for development in Java. It is a programming environment for Java that helps students with its powerful visualization features like source code multiple views, lower level objects and higher level visualizations. As jGRASP is used widely in changing environment [33], it is essentially a program visualization tool, although the object viewers have the potential to approach the functionality of an algorithm visualization tool. It supports the viewing, presenting and constructing engagement levels [95]. The toolbars and interactive design of jGRASP is shown in the figure below.

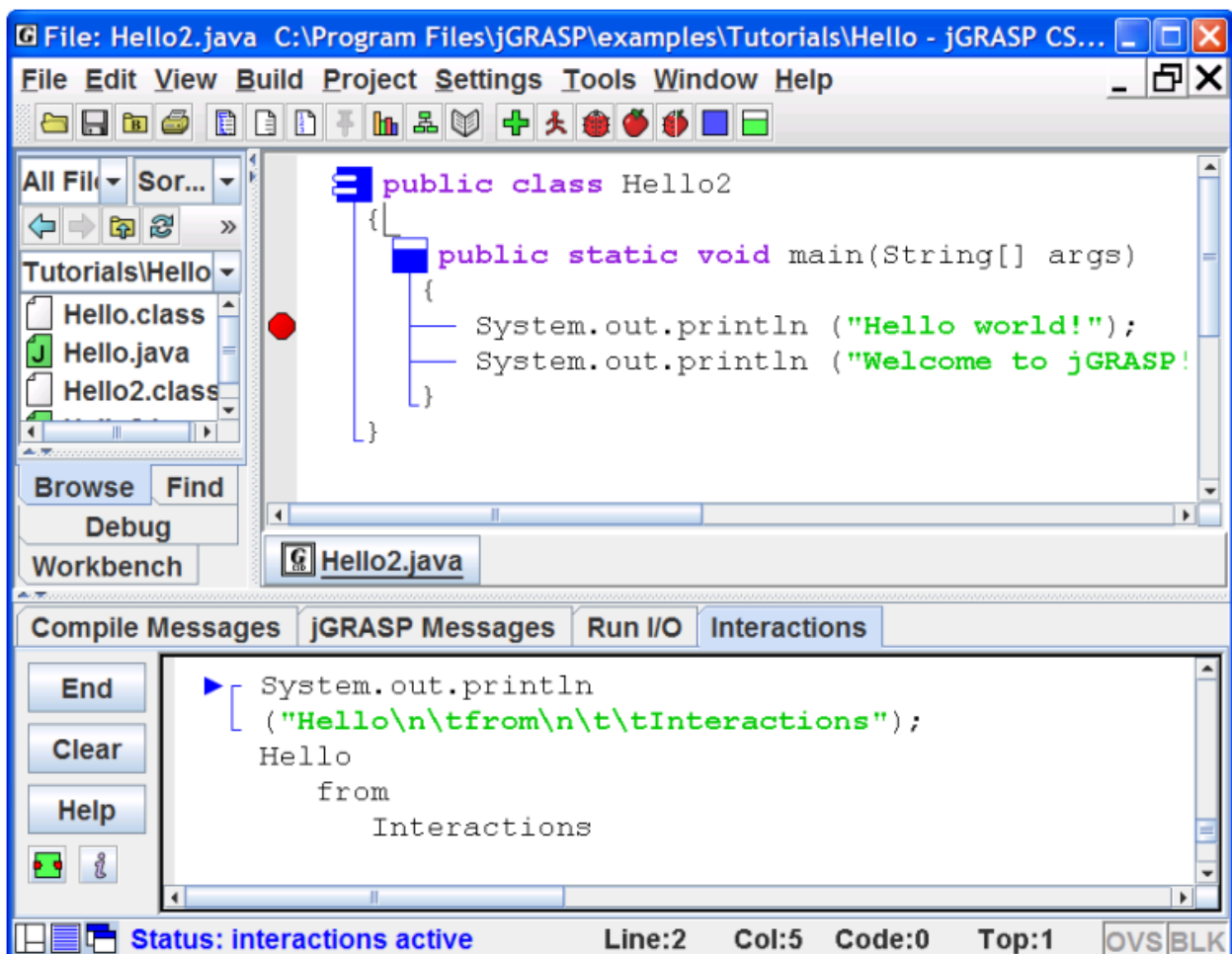


Figure 5.4: Simple representation of jGRASP tool

jGRASP is developed by the GRASP (Graphical Representation of Algorithms, Structures and Processes) research team at Auburn University as a lightweight development environment. The main idea behind the development is to enhance the understanding and clarity of software by supporting automatic generation of software visualizations. It is implemented in Java and can thus be deployed on all the platforms that have a JVM (Java Virtual Machine) of 1.5 version or higher. It supports the generation of Control Structure Diagrams (CSDs), UML class diagrams, Complexity Profile Graphs (CPGs) and has distinct features like viewing objects dynamically integrated with debugger and workbench for Java. These object viewers have mechanism for showing the objects behavior that represent general data structures like stacks, queues, linked lists, binary trees, hash tables etc.. The latest version includes the feature to interpret statements using Interactions window for Java. All these innovative and interactive features are shown in the figure below.

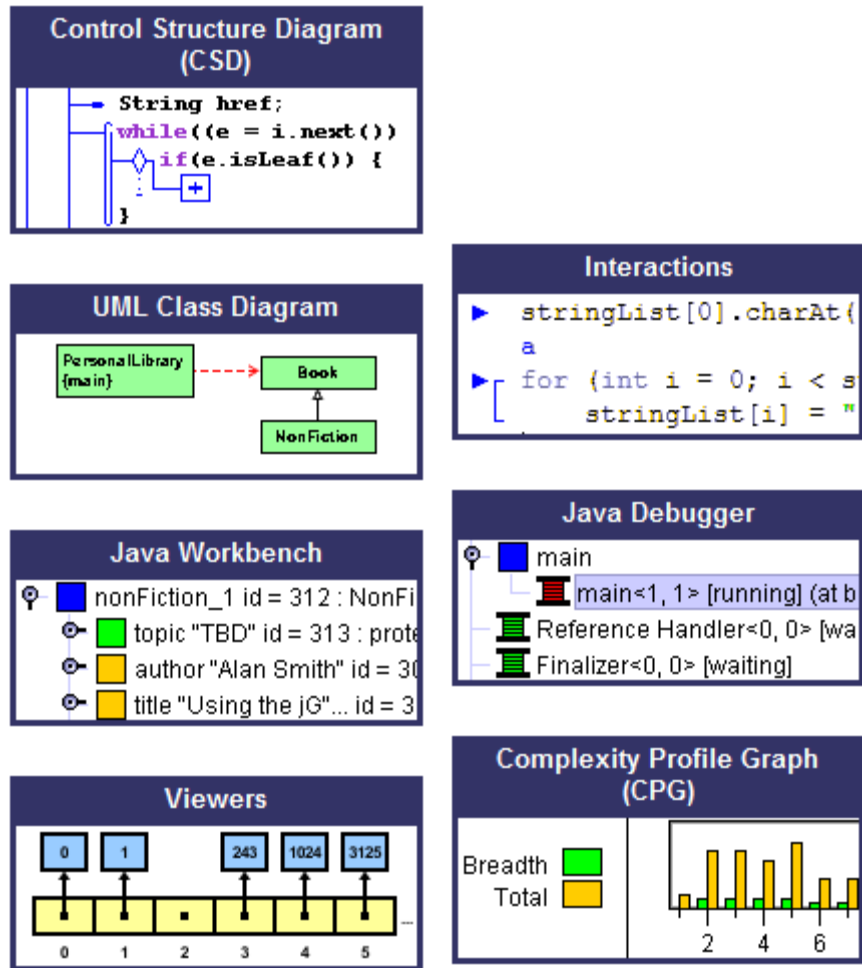


Figure 5.5: Representation of jGRASP features

The *Control Structure Diagram* (CSD) is generated for various languages like Ada, C, C++, Java, Objective-C, and VHDL as an algorithmic level diagram. It helps the user to understand the comprehensibility of programs by clearly interpreting the behavior, paths and structure of each control unit. The *UML class diagram* is automatically generated for the programs written in Java using its class and jar files that helps in understanding its object-oriented behavior. The programmer can generate *dynamic viewers* for primitive types and objects that helps in visualizing a program in steps while in debug mode or when methods are invoked

for an object on the workbench. This *object workbench* along with CSD, UML class diagram and interactions helps the user to invoke methods from instances created in workbench. The *integrated debugger* allows the user to understand each execution step, view the call-stack and local variables data. *Interactions* feature is newly added in the tool where the users have the feasibility to execute their own Java statements and expressions.

5.4 JELIOT

JELIOT is another program visualization system for helping students learn and understand introductory computer science programs [38]. This animation system represents the behavior of a program graphically. Its main aim is to make the novices better understand the in-depth concepts of algorithms and data structures by following the control paths, variable assignments etc. Such concepts are tough to understand while reading through the code using static representation. Various experiments [10] have proved this animation system a concrete tool that augments the grasping capability of a student in understanding the logic and the abstract behavior of software.

The Jeliot program animation system is designed in Java and more widely used for teaching Java programs. It was originally developed by the research team at Helsinki University. Different versions of this system have been released over past ten years that include Eliot, Jeliot I, Jeliot 2000 and Jeliot 3 in the order of their release. The main idea of developing Jeliot system is to actively engage students in the construction of programs and have the students simultaneously understand the execution and behavior of programs through visual representations. This helps them to build a mental model of the software that can be used to know new things and improve the vocabulary about programming concepts.

The user interface has two main panels describing source code used for animation in one of the panels and the corresponding animation on the other. It also has the control toolbar at the bottom of the frame that controls the execution of the program. The output generated by the program can also be viewed dynamically on the lower right of the tool. The user can create constants and also manage the execution speed of the animation. The animation is directly constructed from the Java program code without any additional effort from the user. Many dynamic features like memory allocation, code highlighting, variables loading and storing, evaluation of Java expressions, method calls and control statements can be visualized continuously. These features can be seen from a screenshot of Jeliot below that is captured during the animation execution.

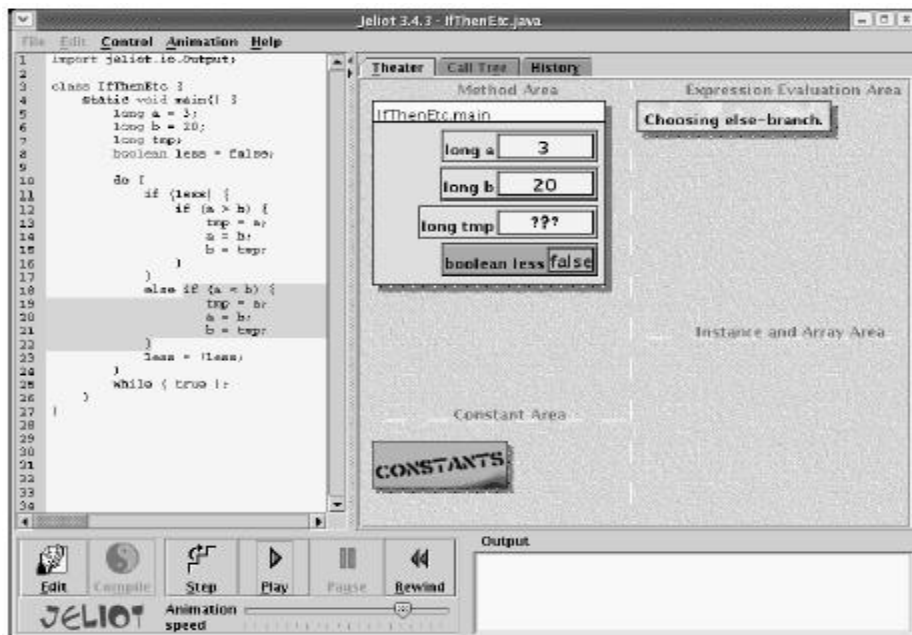


Figure 5.6: Representation of Jeliot System

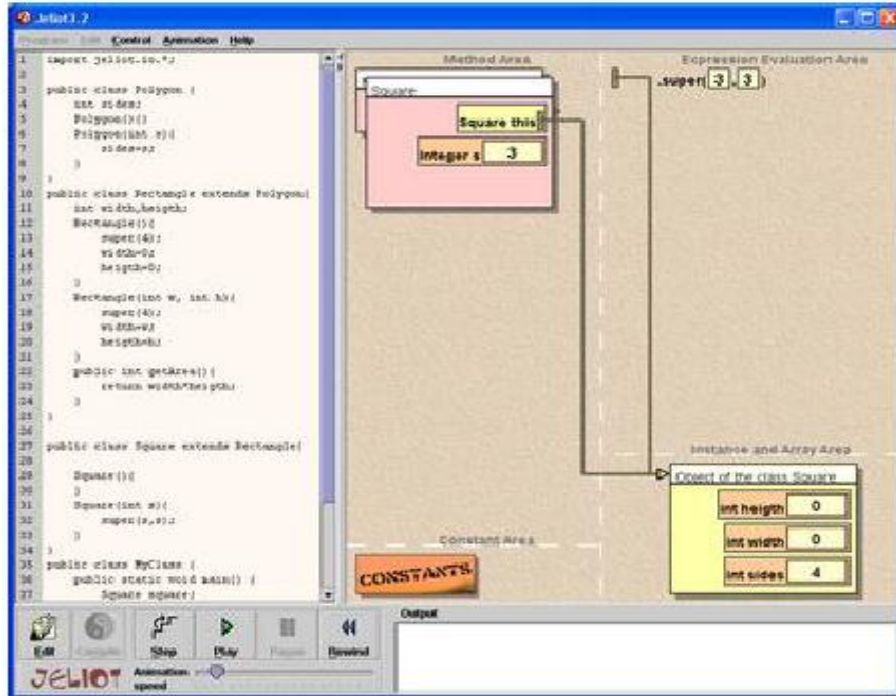


Figure 5.7: Representation of JELIOT 2000 System with more views [45]

Several experiments were conducted for a year to see the effect of Jeliot system on the learning of beginner students. Based on the results from [38], students had better understanding of the concepts, control flow and call-stack by using the vocabulary provided by Jeliot for explanations, than compared to those who have not used the animation system for learning. Another study [79] demonstrated that the Jeliot animation system has increased its capability to grab the attention of students and retain it by using various characteristics.

Ronit Ben-Bassat Levy, during her research has showed that the use of Jeliot animation system enhances the student’s learning of programming and concepts [38]. In her further research, she has studied over the reasons for teachers not accepting it as an efficient tool to communicate its use to the students and then proved the improved acceptance of this tool by teachers. Ebel and Ben-Ari have proved the improved acceptance of Jeliot by students in their

research as it has well-designed acceptance-directing features that attracts student's attention [79].

5.5 MATRIXPRO

As we have seen, many algorithm animation systems have been developed over past three decades. However, most of them are still considered as research prototypes and almost none of them have achieved wide acceptance by instructors to use them in the classrooms. The key reason behind this has been that it takes lot of time and effort for the teachers to understand and develop animations. MatrixPro is developed to simplify this laborious work for the teachers in which they can generate animations based on visual algorithm simulation. The procedure of producing algorithm animations by directly manipulating available library data structures without the need to code anything is termed as visual algorithm simulation [89]. The user has the option to graphically invoke the operations from library that are readily available in order to simulate the behavior and execution of real algorithms. As this tool is equipped with pre-loaded operations, it can understand the semantics of user's operations that greatly helps the instructors to explore the behavior of algorithms with various data input sets and simultaneously work with different scenarios like "what-if" questions that the students can ask in labs or lectures. This approach of the system design can motivate instructors or students to use the tool efficiently when compared to regular pedagogical tools for classroom demonstration. The main focus of MatrixPro is to show and manipulate algorithms and data structures "on-the-fly" in a classroom without any prior preparation by the lecturers. The figure below shows an illustration of red-black tree using this system.

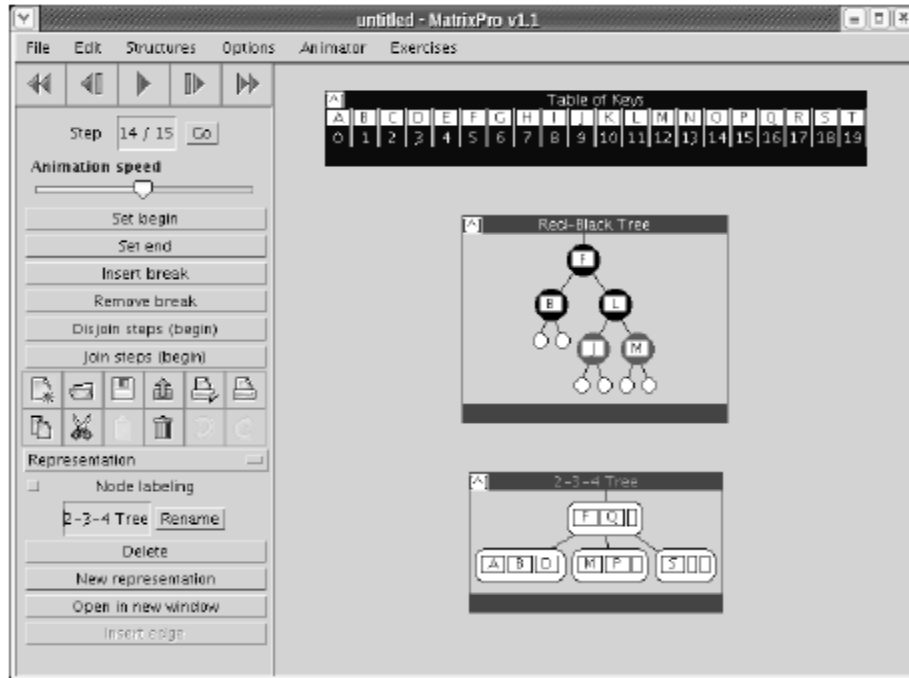


Figure 5.8: Red-Black Tree illustration in MatrixPro

MatrixPro is developed based on the Matrix *algorithm simulation application framework* [90], that provides the basic animation and visualization features for the tool. It has a toolbar and menu bar that share the functionalities of GUI like inserting structures and animation control and modification. The main window has the visualization area where the user can interact with it to understand the execution. The important feature of this tool is its *ex tempore* usage where the system can be used on-the-fly basis. It gives the user an option to apply automatic animation or construct animation using algorithm simulation by hand. It includes the support of custom input data sets by dragging elements from one data structure and dropping into another one. It also allows customization of the level of visualized execution history shown while looking for animation sequence. Using MatrixPro, customized animations can be stored and retrieved for later use. The user can change the menu bar and the pop-up menu options, just by changing the configuration file. MatrixPro has a library of pre-defined data structures that can be used to work

on animations. It includes a set of exercises where it compares the user generated simulation sequence with that of the actual algorithm and gives feedback on it.

CHAPTER 6

PROBLEM STATEMENT

As discussed previously, the field of algorithm visualization has been undertaking a lot of research in order to increase the use of visualization systems among different kinds of users. It is known to be educationally effective when it is made highly interactive [29]. Thus, researchers are trying to make effective visualizations systems for the users so that they can easily build, learn or teach the abstract concepts of computer science algorithms and data structures. But the current visualization systems do not allow great flexibility for the educators to develop animations without spending much time and they do not have enough time to understand the system and build the animations [11]. Furthermore, the availability of ready-made, highly recommended, good quality animations that can be used for teaching is less [28].

6.1 Motivation and associated research questions

Several attempts have been made over the past three decades to use animations widely in computing education. But the results have been underwhelming. Having worked with jGRASP for about two years now, my interest in researching the reason behind these disappointing results has grown. When the users search for animations, they look for the best ones that not only can teach the concepts well but also use advanced visualization system having most of the best features in it. This can help the users interact well with the system and simultaneously understand the algorithm execution. This provoked me to research the literature to know what

they consider as the prominent practices of visualization systems and see if these best practices are incorporated in today's visualization systems. This could be one of the reasons for the inconsistent usage and declining interest of visualization systems among the users. So my research has started to see if the current visualization systems have all the needed requirements and hence my thesis addresses the question whether the currently used algorithm and data structure visualization systems adhere to these practices advocated in the literature.

The primary question that my research addresses is the extent to which current, commonly available algorithm and data structure visualization systems adhere to these best practices advocated in the extant literature. The hypothesis that my research investigates is:

Hypothesis: *Most current visualization systems do not support most best practices.*

The primary thesis question and the research hypothesis are supported by four specific research questions.

RQ1: *To what extent do current visualization systems adhere to the best practices?*

RQ2: *Which best practices are most commonly supported by current visualization systems?*

RQ3: *Is there a set of visualization systems that, as a group, adhere to more best practices than visualization systems in general?*

RQ4: *What portion of visualization systems are recommended for use in the classroom?*

6.2 Methodology

After an extensive review of the literature, nine characteristics are determined to be the best common practices that are categorized based on the type of users and the usage, domain, need, task and resources. They are listed below:

- 1) Need to support flexible execution control of the algorithm visualization
- 2) Visualizations prompt users for answers to questions, predictions etc.
- 3) Provide a context for users to interpret the visualization
- 4) Provide multiple views or representations
- 5) Allow user-specified data sets
- 6) Provide underlying source code and program execution
- 7) Can be used as a lecture aid
- 8) Can be used for self-study
- 9) Can be used as a debugging aid

These best practices are used to evaluate current algorithm and data structure visualization systems and understand the extent to which these systems adhere to the best practices defined.

CHAPTER 7

DATA COLLECTION, ANALYSIS, AND RESULTS

7.1 Use of Algoviz.org

The Algoviz portal (www.algoviz.org) provides information to users and developers of algorithm visualizations through a software environment that is much like a digital library. To make it more effective, both users and developers can review, discuss, and rate the content of this portal so that the instructors are likely to use algorithm visualization more efficiently than the regular practice. The primary focus of the Algoviz portal is to provide the collected and organized data from various educational communities to users through an online tool where the users will have access to the videos and animations to illustrate the algorithm visualizations in addition to the other sources of information. Furthermore, marketing the content of the Algoviz portal to users through social networking sites helps to keep up the communication with a larger number of users in an innovative way. The hope is that by increasing the number of ways to communicate and interconnect with the users, the contribution to the community will be improved. This will help in broadcasting the community-driven content to the users and developers through the Algoviz portal.

The Algoviz portal is an excellent repository of algorithm visualization information. The key informational resource at the Algoviz portal is the *Algorithm Visualization Catalog*, which has hundreds of algorithm visualizations developed by designers using various visualization tools. All these are evaluated and rated based on their use as teaching aid, self-learning or

debugging aid, as shown in the recommendation section of figure below from the AlgoViz catalog.

Algorithms In Action - 2,3,4 Tree

Link(s)	http://ww2.cs.mu.oz.au/aia/
Topic(s)	2-3-4 Tree, B-Tree
Recommendation	
Lecture Aide	Recommended
Self-study Supplement	Recommended
Standalone	Recommended
Debugging Aide	Recommended
Works?	Yes
Delivery Method(s)	Java Applet
Project	Algorithms In Action
Project Relationship	Part of project
Language(s)	English
Author(s)	Linda Stern, Lee Naish, Harald Sondergaard
Institution(s)	University of Melbourne
Activity Level(s)	Animation, Canned data, Random data, Step control
Source Code License	Available but unlicensed
First Published	N/A
Last Modified	2000
Awards	AlgoViz.org Award Nominee - 2010

Figure 7.1: Representation of recommended rating for a given visualization

In addition to the above features, AlgoViz also offers the community of developers to have a connection with the users in order to develop more useful algorithm visualizations. In order to reduce the problems and the present challenges in building up algorithm visualizations, it is important to have a reachable resource through this portal where developers and users can

communicate. To facilitate this feature, the AlgoViz portal provides a complete collection of links to algorithms through AV Catalog where both users and developers can view the topics in the catalog, submit new algorithm visualization, and also browse this catalog provided by many options. The algorithm visualization will be categorized as either recommended, not recommended or has potential and also this particular section will give the user brief information related to topic name, activity level, and delivery method as shown in figure 7.2. By having this information, the user can filter out algorithm visualizations according to the requirement.

Home » Catalog Entry

AV Catalog

Search Content Users

Enter your keywords: Search

Retain current filters

★★★★★	2-3-Tree Animation	Not Recommended
This 2-3 tree is mislabeled as a B-tree. You can't even see what procedure the ...		
Good For: N/A Delivery Method: Java Applet Activity Level: N/A Topic: B-Tree, Search Structures		
★★★★★	A visual implementation of Fortune's Voronoi algorithm	Recommended
Animation of Fortune's beach front algorithm to compute the Voronoi map of a collection of ...		
Good For: N/A Delivery Method: Java Applet Activity Level: Animation, Canned data.. Topic: Delaunay Triangulation..		
★★★★★	A Huffman	Not Recommended
Downloadable Java visualization of encoding using a Huffman tree. It's unclear how the Huffman tree itself ...		
Good For: N/A Delivery Method: Java Application Activity Level: N/A Topic: Huffman coding..		
★★★★★	Algorithms In Action - 2,3,4 Tree	Recommended
Demonstrates building a particular variant of a 2,3,4 Tree (B-Tree of order 4). This AV is specific to 2,3,4 ...		
Good For: N/A Delivery Method: Java Applet Activity Level: Animation, Canned data.. Topic: 2-3-4 Tree..		
★★★★★	Algorithms In Action - Heapsort	Recommended
The applet launches multiple windows. The Explanation window gives a brief description of the algorithm.The ...		
Good For: N/A Delivery Method: Java Applet Activity Level: Animation, Canned data.. Topic: Heapsort..		

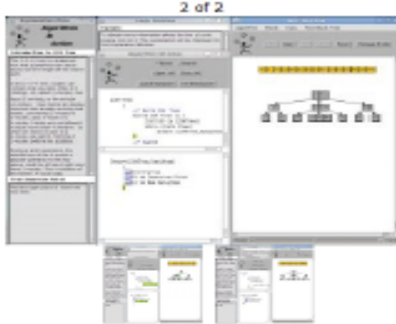
Figure 7.2: Representation of Algoviz Catalog

After selecting particular algorithm visualization it is further organized with more information like the description, evaluation, usage notes, references, ratings, screenshots and videos as shown in figure 7.3. The idea of choosing any topic or feature based on substantiation

of data and not by assumption definitely helps developers and educators to improve the quality of algorithm visualizations. By sharing this portal as a common platform a developer can find valuable comments from forums and can get reports from both fellow developers and users about the working condition and drawbacks of the available algorithm visualizations.

Algorithms In Action - 2,3,4 Tree

Topic(s)	2-3-4 trees, B-trees, Search Structures
Link(s)	http://www2.cs.mu.oz.au/bia/demoindex.html Report broken link
Recommendation	Recommended
Delivery Method(s)	Java Applet
Project	Algorithms In Action
Project Relationship	Part of project
Language(s)	English
Author(s)	Linda Stern, Lee Natshy, Harald Sondergaard
Institution(s)	University of Melbourne
Activity Level(s)	Animation, Canned data, Random data, Step control
AV is good for	Lecture aid, Self-study
Source code is	Available but unlicensed
Last Modified	



DESCRIPTION

Demonstrates building a particular variant of a 2,3,4 Tree (B-Tree of order 4). Note that this demonstration is rather specific to 2,3,4 Trees, and can't really be viewed as a "typical" instance of how B-trees work in general. In particular, the rules for pre-splitting internal nodes on the way down during an insertion are specific to the 2,3,4 Tree.

EVALUATION

Sophisticated use of pseudocode, that can expand to show more or less detail. Has an explanation window for how the data structure works. Slightly limited in that it doesn't support delete, but does a good job at what it does support. As an extra bonus, you can pop open another window that shows the corresponding Red Black Tree (though with no pseudocode or explanation).

USAGE NOTES

The link above takes you to the AIA demonstration index page. Click on the link to the desired AV, and it should load in your browser as a multi-paned Java applet. Note that the level of detail shown in the visualization is directly tied to the level of detail that you choose to expose in the pseudocode pane.

REFERENCES

RATING

★★★★★ Your rating: ☐ ★★★★★

Figure 7.3: Representation of fields for a visualization entry

Another important aspect of the AV Catalog is its collection of URLs or links to algorithm visualizations online and therefore this is the base of the catalog entry structure where

each catalog entry channel contains important information about the visualization such as delivery mechanism, details of the developers, description, and an evaluation of the pedagogical value of the algorithm visualization. To express and report the experiences of the instructors with specific algorithm visualizations in specific course settings, the AlgoViz portal provides a mechanism called the field report that is shown in figure 7.4. Also these field reports will be useful for educators or developers to get feedback from the fellow instructors, teaching techniques and also helps in writing conference papers.

Field Reports

Field Reports are meant to give instructors an opportunity to report actual experiences with specific AVs in specific course settings.

[Submit new Field Report](#) | [Field Reports Forum](#)

Author		Visualizations Used				
<input type="text"/>		Contains any word <input type="text"/>		<input type="button" value="Apply"/>		
Enter a comma separated list of user names.						
	Author	AVs used	Teaser	Posted	Last updated	Views
	shaffer	Interactive Hashing Tutorial	This tutorial was used as part of a quasi-experiment to compare the pedagogical effectiveness of the tutorial as compared to standard textbook/...	2009-05-27 22:13	2010-06-27 10:17	1,039
	rodger	JFLAP	In this lecture, students first learn about pushdown automata. Then the instructor uses JFLAP...	2009-08-09 16:10	2010-06-27 10:18	788
	rodger	JFLAP	In this lesson, we first introduce the formal definition of a deterministic finite automaton (DFA)...	2009-08-09 16:40	2010-06-27 10:18	409
	rodger	JFLAP	This lesson shows how deterministic finite automata (DFA), pushdown automata (PDA) and...	2009-08-09 17:01	2010-06-27 10:18	1,311
	rodger	JFLAP	L-systems are a different type of grammar than the formal language grammars in an automata theory course, so they are good for students to see...	2009-08-09 17:16	2010-06-27 10:18	506
	rodger	JFLAP	This lesson describes Turing machines formally and then uses JFLAP to interact with them. The lesson starts by...	2009-08-09 17:30	2010-06-27 10:20	644
	naps	Deadlock detection algorithm	The AV engagement taxonomy specifies four active levels of learner engagement with AV...	2009-08-10 15:07	2009-08-10 15:07	1,171

Figure 7.4: Field Reports entry in the AlgoViz portal

Another major resource of AlgoViz portal is its Annotated Bibliography which lists over 500 publications based on the research work related to algorithm visualization.

Annotated Bibliography

List Filter

Search the bibliography

Author | Title | Type | Year ▲

Export 527 results: RTF Tagged XML BibTeX

1974

Hopgood, F. R. A., "Computer Animation used as a Tool in Teaching Computer Science", *Proceedings of the IFIP Congress*, pp. 889-892, 1974. RTF Tagged XML BibTeX Google Scholar

1975

Baecker, R. M., "Two systems which produce animated representations of the execution of computer programs", *Proceedings of the fifth SIGCSE technical symposium on Computer science education - SIGCSE '75*: ACM Press, pp. 158 - 167, 1975.

RTF Tagged XML BibTeX Google Scholar

1979

Shrout, P. E., and J. L. Fleiss, "Intraclass correlations: Uses in assessing rater reliability", *Psychological Bulletin*, vol. 86, no. 2, pp. 420-428, 1979. RTF Tagged XML BibTeX Google Scholar





Figure 7.5: Representation of bibliography collection in AlgoViz portal

The collection of curate links to research literature on topics related to algorithm visualizations as shown in figure 7.5 is the Annotated Bibliography index page. As a reference to both the users and developers who intend to develop and explore more on algorithm visualization techniques will have great opportunity to know and learn about the existing technologies from referenced publications provided in this section.

To incorporate one of the other key factors of keeping the communication channel between the learner and expert, AlgoViz portal provides a resource called Forums. This section emphasizes on general discussion on algorithm visualization related topics, educator's forum which helps in teaching techniques in the class rooms, developers forum to pertain discussions related to programming and software development issues, and field reports. The forums overview pages in figure 7.6 shows the title of each thread, number of replies, and number of views with last updated information as well.

Forums

Mark all forums read

Forum	Topics	Posts	Last post
 General Discussion For AV- and site-related topics that do not fit in the other forums.	5	30	Proposal ... by pilucrescenzi 2010-09-28 13:11
 Educators' Forum Discuss using algorithm visualizations as teaching aids in the classroom, as well as teaching about algorithm visualizations.	6	38	2011 ... by shaffer 2010-12-21 13:34
 Field Reports Field Reports are meant to give instructors an opportunity to report actual experiences with specific AVs in specific course settings. You can also browse the field reports by author, post date, and AVs used.	15	25	Use of ... by shaffer 2010-11-11 15:51
 Developers' Forum Discuss the technical and development aspects of AVs, such as programming languages, tools to use, design patterns, etc.	2	3	Gonna make ... by pauloppenheim 2010-03-31 04:25




 Forum Contains New Posts  Forum Contains No New Posts  Forum is Locked

Figure 7.6: Snapshot of Forum discussions in AlgoViz portal

The requirements to build and develop new effective algorithm visualizations are constantly increasing and it is important to make this information available to the users, developers, and researchers. The algorithm visualization community comprised of both experts and learners who come together and contribute to the collective knowledge through the AlgoViz portal, benefits computer education in general. Specifically for this research, however, all the essential features required for the study and development of algorithm visualizations are available in AlgoViz in a well-organized manner. Thus, the AlgoViz portal served as the data collection source for this thesis.

7.2 Summary of the data collection process

As discussed in the Section 7.1, the data to address the primary thesis question and the supporting four research questions are collected from Algoviz.org. The complete information and details about the AlgoViz portal are mentioned in the above Section 7.1. In this section, the process of the data collection is explained in detail.

7.2.1 Data collection process

As the AlgoViz portal unifies the existing collection of online resources to most available algorithm visualizations under the same portal, this feels to be the best repository to collect the data needed for this research. Significant effort has been made to catalog as many existing software visualization systems as possible concentrating in the field of algorithms and data structures. As analysis on large set of data tends to give more accurate results, unique visualization systems with mostly different visualizations are collected. The installation and implementation of each of these systems with a given visualization, has given detailed knowledge about its characteristics and response towards the algorithm. Each of the systems is properly examined and evaluated against the nine best practices collected from the literature. There is a huge collection of links to algorithm visualizations, using which any selected visualization can be redirected. The Description and Usage Notes tags of the visualization also help the user in running the system and the AV with no difficulty. The execution of each visualization resulted in the evaluation of the system against the first six dimensions out of the nine best practices categorized. This can be viewed in the figure 7.7 below for a sample AV system ANIMAL on the Linked Lists data structure.

Animal - Linked Lists

Link(s)

<http://www.animal.ahrgr.de/showAnimationDetails.php3?anim=17>, <http://www.animal.ahrgr.de/showAnimationDetails.php3?anim=36>

Topic(s)

List, Linear Structures

DESCRIPTION

Tutorials describing insertion and deletion for singly and doubly linked lists. Presentation is as a "slideshow," as in a presentation for class.

USAGE NOTES

For detailed instructions on how to install Animal and run Animal AVs, see: <http://www.algoanim.info/Animal2/?q=node/290>. Once you have installed the Animal .jar file and downloaded/unpacked the .zip file of Animal animations, you are now ready to run Animal. Run the .jar file to start Animal. Then go to the "Open" menu item, and browse to where you put the animal animations you got in the .zip file. Pick this AV from the list. You can then step through the animation, or use "kiosk mode" to have the steps fed to you at a constant pace.

Figure 7.7: Description of a particular visualization in AlgoViz

The data for the remaining three practices is collected based on the 'recommendation' rating for a given visualization in the AlgoViz portal. Recommendation of particular algorithm visualization is considered as its overall assessment that is given by the AlgoViz wiki project editors, managers and other raters. This category is divided into four aspects: Is this AV suitable to be used as a Lecture Aid, or Self-study Supplement, or Standalone treatment of that topic, or a Debugging Aid. These aspects are described in Section 7.1. The last three of nine best practices are dealing with the AV rating on Lecture Aid, Self-study Supplement and Debugging Aid. Based on the recommendation rating of visualization in each of the currently available software visualization systems collected, data for the last three practices is collected and analyzed for further evaluation. An example recommendation rating is shown in the figure below.

Recommendation

Lecture Aide	Recommended
Self-study Supplement	Recommended
Standalone	Recommended
Debugging Aide	Not Recommended

Figure 7.8: Recommendation rating based on various categories for a visualization entry

After the entire data set was collected for currently available software visualization systems against the best practices in a spreadsheet, it was analyzed to address the primary thesis question and related research questions.

7.3 Summary of the data collected

Through the process described in the Section 7.1, the data needed for the analysis is collected. Extensive research is performed to gather a significant number of currently used algorithm and data structure visualization systems, result of which is the collection of 30 systems from the AlgoViz portal. These 30 systems are individually examined and a detailed analysis about their functionality is performed. Each of those is examined with an example algorithm using the links provided in Algoviz.org. The algorithm visualizations required for the research are chosen based on the overall recommendation shown in the catalog against each entry as shown in the figure below from Algoviz.org catalog for ANIMAL – Linked Lists. This overall rating is of three types: ‘Recommended’, ‘Not Recommended’, and ‘Has Potential’. All the algorithm visualizations chosen to run on the 30 visualization systems are ‘Recommended’ overall rating. The data is selected this way to run the analysis on today’s highly recommended visualizations to see if they adhere to the best practices defined.

★★★★★	Animal - Linked Lists	Recommended
Tutorials describing insertion and deletion for singly and doubly linked lists. Presentation is as a "slideshow," ...		
Good For: N/A	Delivery Method: Animal Animation	Activity Level: Animation, Canned data...
Topic: List, Linear Structures		

Figure 7.9: Overall recommendation rating for a visualization entry

The data collected is listed in a spreadsheet where each software visualization system is associated with algorithm visualization and the corresponding website link from Algoviz.org. Each visualization system is evaluated based on the nine best practices described in Chapter 4, and have 1-9 dimension numbers associated with them. The data collected is listed in two tables below (Table 7.1 and 7.2) that have to be coupled together to understand the data significance. The dimension number that each visualization system is measured with in table 7.1 is described in with one of the nine practices in table 7.2. The visualization systems are rated based on two factors: YES or NO. A ‘YES’ on the cell is that the corresponding software visualization system supports the given best practice and ‘NO’ is the vice-versa of it. The entire data collected is provided in the Appendix A. Only part of the data is shown in the Table 7.1 below. But the research and analysis for the hypothesis including other related research questions is performed on the entire data collected.

	A	B	C	D	E	F	G	H	I	J	K	L
1							DIMENSION #					
2	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	ALGOVIZ WEB LINK	1	2	3	4	5	6	7	8	9
3	Animal	Linked Lists	http://algoviz.org/catalog/entry/548	YES	NO	YES	NO	NO	NO	YES	YES	NO
4	jGrasp	Linked Lists	http://algoviz.org/catalog/entry/1828	YES	NO	YES	YES	YES	YES	YES	YES	YES
5	JHave	Binary Search Tree	http://algoviz.org/catalog/entry/895	YES	YES	YES	YES	YES	YES	YES	YES	NO
6	AlViE	Longest Common Subsequence	http://algoviz.org/catalog/entry/1921	YES	NO	YES	YES	YES	YES	YES	YES	YES
7	Trakla	AVL Tree	http://algoviz.org/node/1197	YES	NO	YES	YES	NO	NO	YES	YES	NO
8	Jeliot 3	Priority Queue	http://algoviz.org/catalog/entry/1824	YES	NO	YES	YES	YES	YES	YES	YES	NO
9	CS Animated	Red-Black Tree	http://algoviz.org/catalog/entry/864	YES	NO	YES	NO	NO	NO	YES	YES	NO
10	Algorithms In Action	Quick Sort	http://algoviz.org/catalog/entry/530	YES	NO	YES	YES	YES	YES	YES	YES	NO
11	Data Structures Navigator	Merge Sort	http://algoviz.org/catalog/entry/739	YES	NO	NO	NO	NO	YES	YES	YES	NO
12	Uuhistle	2,3,4 Tree	http://algoviz.org/catalog/entry/1880	YES	NO	YES	YES	YES	YES	YES	YES	YES
13	Data Structure Visualization	Dijkstra's Algorithm	http://algoviz.org/catalog/entry/749	YES	NO	NO	NO	YES	NO	YES	YES	NO
14	Interactive Data Structure Visualization	Graph Search	http://algoviz.org/catalog/entry/800	YES	NO	NO	NO	NO	NO	YES	YES	NO
15	Auckland	Huffman Coding	http://algoviz.org/catalog/entry/619	YES	NO	YES	YES	NO	YES	YES	YES	NO
16	Animator	Quick Sort	http://algoviz.org/catalog/entry/602	YES	NO	YES	YES	NO	YES	NO	NO	NO

Table 7.1: Representation of the data collected based on the AlgoViz portal

	A	B
1	DIMENSION #	PRACTICE
2	1	Users can interact with visualization by controlling animation steps, start/stop, speed, etc.
3	2	Visualization prompts users for answers to questions, predictions, etc.
4	3	Provides a context for users to interpret the visualization (text description, source code etc.)
5	4	Provides multiple views or representations
6	5	Allows user-specified data sets.
7	6	Provides underlying source code and program execution (e.g., statement highlighting, call-stack).
8	7	Can it be used as a lecture aid?
9	8	Can it be used for self-study?
10	9	Can it be used as a debugging aid?

Table 7.2: Representing best practices as stated in this thesis

The data collected here is the core information for examining the visualization systems and to evaluate them based on different types of analysis performed to address various research questions. This is explained in detail in the Section 7.4.

7.4 Data analysis

To address the primary thesis question: ‘To what extent do current, commonly available visualization systems adhere to the best practices advocated in the literature?’ a hypothesis has been proposed saying that most current software visualization systems do not adhere to most the best practices reported in the literature. Series of steps have been performed to address the problem in order. It comprises of literature review in the beginning that involves gathering much information about the respective topic. Based on the research of the literature, the common best practices are sorted out, that are essential for any visualization for it to be considered as useful learning material. Several currently available software visualization systems are then evaluated against these best practices and the analytical data is extracted. This is analyzed in depth to address several questions that are discussed below. It helps the developers to know how to

develop high quality algorithm visualizations, the users on how to use them effectively, and the educators on how to find an effective AV.

The data analysis is divided into five sub sections comprising of analysis on the collected data to address the hypothesis and four related research questions (RQs). The overall analysis from these five steps (refer sections below) helps the current research to focus on existing AV problem. Though only part of data collected is shown in the chapter here and entire data in the Appendix, the data analysis is performed on the whole data collected. It helps in achieving more accurate and less precision results from the large dataset analysis.

7.4.1 Data analysis to address the hypothesis

The data collected is employed for analyzing the software visualization systems based on the number of best practices they support. Depending on the results, the hypothesis that most currently available software systems do not adhere to most best practices from literature is assessed. A numerical analysis has been performed on the gathered data to check on how many practices or features are supported by a given visualization system out of the 9 best practices. The statistical data for this is also calculated and all this analytical data is represented in the table 7.3 below. Based on the analysis of 30 different currently available visualization systems from the table below, it is clearly seen that none of these systems support all of the best practices. Only 33% support more than half of the best practices. This refutes the null hypothesis (viz., most visualization systems support most of the best practices), and thus we can claim that our hypothesis is supported by the data.

	A	B	C	D	E
1	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	NUMBER OF BEST PRACTICES SUPPORTED (OUT OF 9)	NUMBER OF BEST PRACTICES NOT SUPPORTED	PERCENTAGE OF BEST PRACTICES SUPPORTED IN A SV SYSEM
2	jGrasp	Linked Lists	8	1	88.9%
3	JHave	Binary Search Tree	8	1	88.9%
4	AlViE	Longest Common Subsequence	8	1	88.9%
5	Uuhistle	2,3,4 Tree	8	1	88.9%
6	Jeliot 3	Priority Queue	7	2	77.8%
7	Algorithms In Action	Quick Sort	7	2	77.8%
8	Auckland	Huffman Coding	6	3	66.7%
9	JFLAP	Mealy Machine	6	3	66.7%
10	Virginia Tech AV	Heap Sort	5	4	55.6%
11	Trakla	AVL Tree	5	4	55.6%
12	Animal	Linked Lists	4	5	44.4%
13	CS Animated	Red-Black Tree	4	5	44.4%
14	Data Structures Navigator	Merge Sort	4	5	44.4%
15	Data Structure Visualization	Dijkstra's Algorithm	4	5	44.4%
16	Animator	Quick Sort	4	5	44.4%
17	JCAT	Heap Sort	4	5	44.4%
18	Kovac's Tree Project	B Tree	4	5	44.4%
19	OLLI	Heap Sort	4	5	44.4%
20	Smith College AV	Sorting Algorithms	4	5	44.4%
21	Interactive Data Structure Visualization	Graph Search	3	6	33.3%
22	JAVENGA	Shortest Paths	3	6	33.3%
23	Swan	Huffman Coding	3	6	33.3%
24	Ghosh's AV	AVL Tree	2	7	22.2%
25	Java Applets Centre	Binary Search Tree	2	7	22.2%
26	Galgo	Euler Path	2	7	22.2%
27	Gawain	Bubble Sort	2	7	22.2%
28	JAWAA	Shell Sort	2	7	22.2%
29	Upatras	Kruskal Algorithm	2	7	22.2%
30	Sort Algorithm Animator	Insertion Sort	1	8	11.1%
31	UniSorter	Radix Sort	1	8	11.1%

Table 7.3: Data analysis to address the hypothesis

The algorithms and data structures used in the visualizations for analysis are common ones in CS2 and CS3 courses. Also, all these visualizations are recommended by the AlgoViz active committee. Evaluating such visualizations would give more practical and better results. The results are listed in a spreadsheet and shown as a table above (in Table 7.3) with the visualization systems in the decreasing order of the practices supported. The percentage of the best practices supported in a given software visualization system is calculated based on the data analyzed and is tabulated to determine the strength of each system in the current period. For example, the data from the Table 7.3 can be interpreted saying that the jGRASP visualization system supports 8 of the 9 best practices; i.e., it supports 88.9% of the common practices today.

The higher the number of practices supported, the more is the chance of improving the use of visualizations by various types of users (discussed in the section 1.4) and overcoming the existing problems. Depending on the rating provided in the table as percentage, the system or visualization developer could easily understand the flaws in it and would be able to build new solution. More number of effective visualizations would ultimately help the students learn better and improve their understanding of algorithms and data structures.

7.4.2 Data analysis to address RQ1

The current research is assisted with four related research questions that can further address the primary thesis question and the associated the hypothesis. The first of them is to what extent does currently available software visualization systems adhere to the best practices as reported in literature. It is important to understand the extent to which the visualization systems promote learning by supporting the best practices. This research question is addressed by performing quantitative analysis on the data collected from Algoviz.org (shown below) and calculating the overall percentage of the number of commonly available software visualization systems that support a given practice, out of the 9 best practices. For each practice, the percentage of how many systems meet that practice is calculated by counting the number of ‘YES’ fields from the Table 7.4 (same as Table 7.1, but shown again for easy reference with Table 7.5) for each dimension number 1 through 9. The results can be seen from the Table 7.5 while each dimension number is clearly described in the Table 7.6 (same as Table 7.2).

	A	B	C	D	E	F	G	H	I	J	K	L	
1							DIMENSION #						
2	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	ALGOVIZ WEB LINK	1	2	3	4	5	6	7	8	9	
3	Animal	Linked Lists	http://algoviz.org/catalog/entry/548	YES	NO	YES	NO	NO	NO	YES	YES	NO	
4	jGrasp	Linked Lists	http://algoviz.org/catalog/entry/1828	YES	NO	YES	YES	YES	YES	YES	YES	YES	
5	JHave	Binary Search Tree	http://algoviz.org/catalog/entry/895	YES	YES	YES	YES	YES	YES	YES	YES	NO	
6	AlVE	Longest Common Subsequence	http://algoviz.org/catalog/entry/1921	YES	NO	YES	YES	YES	YES	YES	YES	YES	
7	Trakla	AVL Tree	http://algoviz.org/node/1197	YES	NO	YES	YES	NO	NO	YES	YES	NO	
8	Jehiot 3	Priority Queue	http://algoviz.org/catalog/entry/1824	YES	NO	YES	YES	YES	YES	YES	YES	NO	
9	CS Animated	Red-Black Tree	http://algoviz.org/catalog/entry/864	YES	NO	YES	NO	NO	NO	YES	YES	NO	
10	Algorithms In Action	Quick Sort	http://algoviz.org/catalog/entry/530	YES	NO	YES	YES	YES	YES	YES	YES	NO	
11	Data Structures Navigator	Merge Sort	http://algoviz.org/catalog/entry/739	YES	NO	NO	NO	NO	YES	YES	YES	NO	
12	Uhistle	2,3,4 Tree	http://algoviz.org/catalog/entry/1880	YES	NO	YES	YES	YES	YES	YES	YES	YES	
13	Data Structure Visualization	Dijkstra's Algorithm	http://algoviz.org/catalog/entry/749	YES	NO	NO	NO	YES	NO	YES	YES	NO	
14	Interactive Data Structure Visualization	Graph Search	http://algoviz.org/catalog/entry/800	YES	NO	NO	NO	NO	NO	YES	YES	NO	
15	Auckland	Huffman Coding	http://algoviz.org/catalog/entry/619	YES	NO	YES	YES	NO	YES	YES	YES	NO	
16	Animator	Quick Sort	http://algoviz.org/catalog/entry/602	YES	NO	YES	YES	NO	YES	NO	NO	NO	

Table 7.4: Representation of the data collected based on the AlgoViz portal

	A	B	C	D	E	F	G	H	I	J
1		1	2	3	4	5	6	7	8	9
2	Percentage of commonly available Software Visualization Systems that support a given practice	96.67%	3.33%	66.67%	50%	46.67%	50%	53.33%	46.67%	10%

Table 7.5: Data analysis to address research related question 1

	A	B
1	DIMENSION #	PRACTICE
2	1	Users can interact with visualization by controlling animation steps, start/stop, speed, etc.
3	2	Visualization prompts users for answers to questions, predictions, etc.
4	3	Provides a context for users to interpret the visualization (text description, source code etc.)
5	4	Provides multiple views or representations
6	5	Allows user-specified data sets.
7	6	Provides underlying source code and program execution (e.g., statement highlighting, call-stack).
8	7	Can it be used as a lecture aid?
9	8	Can it be used for self-study?
10	9	Can it be used as a debugging aid?

Table 7.6: Representing best practices as stated in this thesis

It is easy to understand the analysis when seen in numbers or in a relative comparison. For example, the data can be interpreted saying that only 46.67% of the currently available

visualizations support the 5th practice which allows user-specified data input while running the visualization. By examining the analysis for this question, it is important to note that none of the practices is supported by all the available visualizations. Or in other words, it can be said that none of the current visualization systems have all the common features or practices (as defined in chapter 4) that make them effective. This could be one of the reasons for the decline in the use of visualizations by the users. But positively, there are several systems supporting most of the best practices (can be seen from the Table 7.4), which makes them more established today when compared to the others.

7.4.3 Data analysis to address RQ2

The second research question related to the hypothesis is which best practices are most commonly present in currently available software visualization systems. It is true that a small number of practices featured by algorithm visualizations have significant impact on its effective usage.

To address this related research question, the analyzed data from Section 7.4.2, Table 7.5 is taken to perform further analysis. The analyzed data has the percentage calculated on how many commonly available software visualization systems support a given best practice.

	A	B	C	D	E	F	G	H	I	J
1		1	2	3	4	5	6	7	8	9
2	Percentage of commonly available Software Visualization Systems that support a given practice	96.67%	3.33%	66.67%	50%	46.67%	50%	53.33%	46.67%	10%

Table 7.7: Data analysis to address research related question 2

	A	B
1	DIMENSION #	PRACTICE
2	1	Users can interact with visualization by controlling animation steps, start/stop, speed, etc.
3	2	Visualization prompts users for answers to questions, predictions, etc.
4	3	Provides a context for users to interpret the visualization (text description, source code etc.)
5	4	Provides multiple views or representations
6	5	Allows user-specified data sets.
7	6	Provides underlying source code and program execution (e.g., statement highlighting, call-stack).
8	7	Can it be used as a lecture aid?
9	8	Can it be used for self-study?
10	9	Can it be used as a debugging aid?

Table 7.8: Representing best practices as stated in this thesis

To see the commonly supported best practices from the Table 7.5, it is clearly the highest percentage value among the nine. As we see here, the most commonly present characteristics in all the currently available algorithms and data structure visualizations are practices 1 and 3 from the Table 7.8. So, most of the existing visualizations have user controls like start/stop, forward/backward direction control, speed etc. to help the users interact with the visualization easily. This number comes up to be 96.67%, which means visualizations developed by 29 systems out 30 support this respective practice. It is good to know that most of the visualization developers understand the importance of user interaction and incorporated interactive designs for better learning. This is the primary step from the visualizations to have an impact on the student's learning. The next practice that the most common visualizations support is that they provide the users a context to interpret the visualizations better. It is 66.67% of the current visualization systems support this feature, which means 20 out of 30 systems from the dataset have this characteristic available for their visualizations. It is exciting to know that 2/3rd of the existing systems have the option to add this feature to their visualizations. The developers need to understand the novice user's perspective and provide efficient resources using the

visualization systems like text description explaining the algorithm, source or pseudo code compatibly shown next to the animation in order to assist the learner’s understanding.

7.4.4 Data analysis to address RQ3

The third research related question is whether any set of software visualization systems that support more number of best practices fall into a certain category or is it whether certain types of software visualization systems more likely to support the 9 best practices. To address this question, data analyzed for the hypothesis is taken from the Table 7.3 and sorted the visualization systems based on the number of best practices supported. The list of systems that have 50% or greater percentage of best practices supported is shown in the Table 7.9 below. While the remaining systems with less than 50% of practices supported are shown in the next Table 7.10. This part of analysis comprises of 3 different categories where the visualization systems supporting more number of best practices come into.

	A	B	C	D	E
	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	NUMBER OF BEST PRACTICES SUPPORTED (OUT OF 9)	NUMBER OF BEST PRACTICES NOT SUPPORTED	PERCENTAGE OF BEST PRACTICES SUPPORTED IN A SV SYSEM
1					
2	jGrasp	Linked Lists	8	1	88.9%
3	JHave	Binary Search Tree	8	1	88.9%
4	AlViE	Longest Common Subsequence	8	1	88.9%
5	Uuhistle	2,3,4 Tree	8	1	88.9%
6	Jeliot 3	Priority Queue	7	2	77.8%
7	Algorithms In Action	Quick Sort	7	2	77.8%
8	Auckland	Huffman Coding	6	3	66.7%
9	JFLAP	Mealy Machine	6	3	66.7%
10	Virginia Tech AV	Heap Sort	5	4	55.6%
11	Trakla	AVL Tree	5	4	55.6%

Table 7.9: Data showing SV systems supporting more practices in order to address research related question 3

1st Category: The systems that possess more practices are jGrasp, JHave, AIViE, Uuhistle, Jeliot, AIA, Auckland, JFlap, Trakla and Virginia Tech AV. Most of these systems from Table 7.9 are well established software visualization groups like jGRASP, JHAVE, TRAKLA, JELIOT, AIViE, Algorithms In Action, Auckland etc. when compared to the list from Table 7.10 that have established groups like ANIMAL, JAWAA. This can be interpreted saying that most of the established groups are likely to exhibit more number of best practices when compared to others from Table 7.10.

2nd Category: When a particular visualization is run using any of these 30 systems, it is clearly noticed from the design view that some systems concentrate exclusively on a certain set of algorithms like sorting algorithms while some of them are designed for general purpose to support any given random algorithm. It can be inferred from here that systems that support a particular set of algorithms tend to support more practices than those designed for general purpose use.

	A	B	C	D	E
1	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	NUMBER OF BEST PRACTICES SUPPORTED (OUT OF 9)	NUMBER OF BEST PRACTICES NOT SUPPORTED	PERCENTAGE OF BEST PRACTICES SUPPORTED IN A SV SYSEM
12	Animal	Linked Lists	4	5	44.4%
13	CS Animated	Red-Black Tree	4	5	44.4%
14	Data Structures Navigator	Merge Sort	4	5	44.4%
15	Data Structure Visualization	Dijkstra's Algorithm	4	5	44.4%
16	Animator	Quick Sort	4	5	44.4%
17	JCAT	Heap Sort	4	5	44.4%
18	Kovac's Tree Project	B Tree	4	5	44.4%
19	OLLI	Heap Sort	4	5	44.4%
20	Smith College AV	Sorting Algorithms	4	5	44.4%
21	Interactive Data Structure Visualization	Graph Search	3	6	33.3%
22	JAVENGA	Shortest Paths	3	6	33.3%
23	Swan	Huffman Coding	3	6	33.3%
24	Ghosh's AV	AVL Tree	2	7	22.2%
25	Java Applets Centre	Binary Search Tree	2	7	22.2%
26	Galgo	Euler Path	2	7	22.2%
27	Gawain	Bubble Sort	2	7	22.2%
28	JAWAA	Shell Sort	2	7	22.2%
29	Upatras	Kruskal Algorithm	2	7	22.2%
30	Sort Algorithm Animator	Insertion Sort	1	8	11.1%
31	UniSorter	Radix Sort	1	8	11.1%

Table 7.10: Data showing SV systems supporting fewer practices in order to address research related question 3

3rd Category: The AV Catalog from Algoviz.org has hundreds of visualizations listed. When this data is sorted based on the recommendation level, the data obtained is as shown in the figure 7.10. The filtered data consists of visualizations developed by various algorithm visualization systems. The visualizations can be either ‘algorithms’ or ‘data structures’. When the filtered data is analyzed and examined, most of the recommended visualizations are different kinds of ‘algorithms’. From this data analysis it can be suggested that visualizations that are focused more on data structures generally exhibit fewer best practices than those focused on a particular algorithm.

AV Catalog

Search Content Users

Enter your keywords:

Search

Retain current filters

Algorithms In Action - 2,3,4 Tree

★★★★★

Algorithms In Action - 2,3,4 Tree

Recommended

Demonstrates building a particular variant of a 2,3,4 Tree (B-Tree of order 4). This AV is specific to 2,3,4 ...

Delivery Method: Java Applet Topic: 2-3-4 Tree..

Algorithms In Action - Heapsort

★★★★★

Algorithms In Action - Heapsort

Recommended

The applet launches multiple windows. The Explanation window gives a brief description of the algorithm.The ...

Delivery Method: Java Applet Topic: Heapsort..

Algorithms In Action - Multiway Radix Trie

★★★★★

Algorithms In Action - Multiway Radix Trie

Recommended

Demonstrates building a multiway Radix Trie. Given a set of values, the trie structure is built ...

Delivery Method: Java Applet Topic: Tries, Search Structures

Algorithms In Action - Quicksort

★★★★★

Algorithms In Action - Quicksort

Recommended

The applet launches multiple windows. The Explanation window gives a brief description of the algorithm.The ...

Delivery Method: Java Applet Topic: Quicksort..

Algorithms In Action - Radix Trie

★★★★★

Algorithms In Action - Radix Trie

Recommended

Demonstrates building a Radix Trie. Given a set of values, the trie structure is built step ...

Delivery Method: Java Applet Topic: Tries, Search Structures

Algorithms In Action - Skip List

★★★★★

Algorithms In Action - Skip List

Recommended

Demonstrates building a Skip List. Given a set of values, the trie structure is built step ...

Delivery Method: Java Applet Topic: Skip list..

ALVIE - Closest Pair

★★★★★

ALVIE - Closest Pair

Recommended

Demonstrates a divide and conquer algorithm for computing the closest pair of points within a specified ...

Delivery Method: Java Application Topic: Computational Geometry

ALVIE - Longest Common Subsequence

★★★★★

ALVIE - Longest Common Subsequence

Recommended

This visualization shows the behavior of a dynamic programming algorithm for solving the longest common subsequence ...

Delivery Method: Flash, Java Application Topic: Bioinformatics, Dynamic programming

Animal - Backtracking (8 Queens)

★★★★★

Animal - Backtracking (8 Queens)

Recommended

Presents a small tutorial on backtracking, and then has a detailed animated example of the 4 Queens ...

Delivery Method: Animal Animation Topic: Backtracking, Algorithmic Techniques

Animal - Dijkstra's Algorithm

★★★★★

Animal - Dijkstra's Algorithm

Recommended

Presents a slideshow with an example walking through Dijkstra's algorithm performed on a small graph.

Delivery Method: Animal Animation Topic: Single-source shortest path problem..

Current search

Search found 66 items

- Catalog Entry
- Recommended
- Recommended
- Recommended

Filter by Projects

Show more

Filter by Language

Filter by Works?

Filter by Delivery Methods

Filter by Source Code Availability and Licenses

Filter by Activity Level

Filter by Standalone?

- Recommended
- Not Recommended (6)
- Has Potential (2)

Filter by Self-study Supplement

- Recommended

Filter by Lecture Aide

- Recommended
- Has Potential (1)

Filter by Debugging Aide

- Not Recommended (54)
- Recommended (11)
- Has Potential (1)

Figure 7.10: Data analysis to address research related question 3

7.4.5 Data analysis to address RQ4

The fourth research related question is how many currently available software visualization systems are recommended for use in the classroom. The primary use of algorithm visualizations is to help learners understand the abstract computer science concepts better. The effectiveness of a given algorithm visualization is measured by its pedagogical use [29]. It builds upon not only on how well students acquire knowledge from the visualizations but also on how widely instructors use them for pedagogical purposes. This question is addressed by analyzing the data collected from the AlgoViz portal. It is sorted based on the 7th practice (as seen from the Table 7.12) which is if the visualization developed by one of the 30 systems collected can be used as a lecture aid or not. The results are calculated by counting the number of systems that can be used for instruction and are shown in the Table 7.11.

	A	B	C
1	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	BEST PRACTICE # 7 - CAN IT BE USED AS A LECTURE AID?
2	Animal	Linked Lists	YES
3	jGrasp	Linked Lists	YES
4	JHave	Binary Search Tree	YES
5	AlViE	Longest Common Subsequence	YES
6	Trakla	AVL Tree	YES
7	Jeliot 3	Priority Queue	YES
8	CS Animated	Red-Black Tree	YES
9	Algorithms In Action	Quick Sort	YES
10	Data Structures Navigator	Merge Sort	YES
11	Uuhistle	2,3,4 Tree	YES
12	Data Structure Visualization	Dijkstra's Algorithm	YES
13	Interactive Data Structure Visualization	Graph Search	YES
14	Auckland	Huffman Coding	YES
15	JFLAP	Mealy Machine	YES
16	JAWAA	Shell Sort	YES
17	Kovac's Tree Project	B Tree	YES
18	Animator	Quick Sort	NO
19	Ghosh's AV	AVL Tree	NO
20	Sort Algorithm Animator	Insertion Sort	NO
21	Java Applets Centre	Binary Search Tree	NO
22	Galgo	Euler Path	NO
23	Gawain	Bubble Sort	NO
24	JCAT	Heap Sort	NO
25	JAVENGA	Shortest Paths	NO
26	OLLI	Heap Sort	NO
27	Smith College AV	Sorting Algorithms	NO
28	Virginia Tech AV	Heap Sort	NO
29	Swan	Huffman Coding	NO
30	Upatras	Kruskal Algorithm	NO
31	UniSorter	Radix Sort	NO

Table 7.11: Data analysis to address research related question 4

	A	B
1	DIMENSION #	PRACTICE
2	1	Users can interact with visualization by controlling animation steps, start/stop, speed, etc.
3	2	Visualization prompts users for answers to questions, predictions, etc.
4	3	Provides a context for users to interpret the visualization (text description, source code etc.)
5	4	Provides multiple views or representations
6	5	Allows user-specified data sets.
7	6	Provides underlying source code and program execution (e.g., statement highlighting, call-stack).
8	7	Can it be used as a lecture aid?
9	8	Can it be used for self-study?
10	9	Can it be used as a debugging aid?

Table 7.12: Representing best practices as stated in this thesis

It is observed that 53.33% of the currently available algorithm visualization systems are recommended for use in the classroom, which means 16 out of 30 systems in the dataset collected from AlgoViz portal support this practice. So it is more than 50% of the existing systems or the visualizations developed by them are capable to be used for pedagogy purposes, which is a positive note. But the impact they have on students is relatively low as they should also be aware of how to use them effectively. It is equally difficult for the instructors too as they would have to spend lot of extra time discovering the working of the system, understanding the visualization, and learning to develop their own visualizations. This issue is discussed in detail in the section 4.7. The developers have to always have students' perspective in mind while creating new visualizations as the outcome needs to be pedagogy oriented.

CHAPTER 8

SUMMARY AND CONCLUSIONS

Software visualization developers and researchers are constantly looking for new approaches to grab students' attention for learning through visualizations and continuously diagnosing the existing problems for better solutions that improves the use of algorithm visualizations and student learning. While some studies indicated improvement in student learning using visualization, others could hardly find differences between traditional teaching and teaching through animation or even had contradictory results. There may be several reasons for this, such as less student engagement in the visualization, inappropriate usage, less use in the classroom, less interaction between the user and the animation etc.

Through this research we were able to clearly quantify the extent to which current visualization systems support best practices from the literature. The results were not satisfactory, as most systems do not exhibit the majority of the pedagogical features that help in learning difficult concepts and improving the ability to understand algorithms and data structures. Indeed, it is noteworthy that none of the 30 systems surveyed supported all nine best practices. Few practices are supported commonly by all the existing visualizations, but the key feature that is supported most widely is the level of engagement, which has higher chances to enhance student learning. Another important feature is whether the visualizations can be used as learning aid. It involves the satisfaction of both instructors and students.

The problems faced by students, teachers or programmers in understanding algorithms are not all solved by visualization technology, of course. Even today, for certain complex concepts in computer science, textual communication proved to be superior and effective when compared to the visual programming languages and the visualizations [51, 52]. However, it is clear that appropriate visualizations, particularly those that exhibit known best practices, can have a positive impact on teaching and learning. The research reported in this thesis is an important step forward in quantifying the maturation of the field. While current systems as a whole are not mature (i.e., do not adhere to most best practices), there are selected groups of visualization systems that are highly mature and have a great potential for positive impact.

REFERENCES

- 1) R. Baecker (1975) Two systems which produce animated representations of the execution of computer programs. *SIGCSE Bulletin* 7, 158-167.
- 2) M. H. Brown (1988) *Algorithm Animation*. The MIT Press, Cambridge, MA.
- 3) J. T. Stasko (1990) TANGO: A framework and system for algorithm animation. *IEEE Computer* 23, 27-39.
- 4) J.T. Stasko (1997) Using student-built animations as learning aids. In: *Proceedings of the ACM Technical Symposium on Computer Science Education*. ACM Press, New York, pp. 25-29.
- 5) C. D. Hundhausen (1999) Toward effective algorithm visualization artifacts: designing for participation and communication in an undergraduate algorithms course. Unpublished Ph.D. dissertation, Department of Computer and Information Science, University of Oregon.
- 6) P. Gloor (1998) Animated algorithms. In: *Software Visualization: Programming as a Multimedia Experience* (M. Brown, J. Domingue, B. Price & J. Stasko, eds) The MIT Press, Cambridge, MA, pp. 409-416.
- 7) J. S. Gurka, & W. Citrin (1996) Testing effectiveness of algorithm animation. In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*. IEEE Computer Society Press, Los Alamitos, CA, pp. 182-189.

- 8) T. Naps (1990) Algorithm visualization in computer science laboratories. In: *Proceedings of the 21st SIGCSE Technical Symposium on Computer Science Education*. ACM Press, New York, pp. 105-110.
- 9) Akingbade, A., Finley, T., Jackson, D., Patel, P., and Rodger, S. H. JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses. In *Proceedings of the 34th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003)*, Reno, Nevada (2003), ACM Press, New York, pp. 162-166.
- 10) Levy, R. B.-B., Ben-Ari, M., and Uronen, P. A. (2000). An Extended Experiment with Jeliot 2000. In *Proceedings of the First International Program Visualization Workshop, Porvoo, Finland* (July 2001), University of Joensuu Press, Finland, pp. 131-140.
- 11) T.L. Naps et al., “Exploring the role of visualization and engagement in computer science education”, ACM SIGCSE Bulletin, Volume 35, Issue 2, ACM, New York USA, 2003, pp. 131-152
- 12) Price, B., Baecker, R., and Small, I. An Introduction to Software Visualization. In *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1998, ch. 1, pp. 3-27.
- 13) Robling, G., and Freisleben, B. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing* 13, 2 (2002), 341-354.
- 14) T.L. Naps et al., “Evaluating the educational impact of visualization”, *Annual Joint Conference Integrating Technology into Computer Science Education*, ACM, New York USA, 2003, pp 124-136.

- 15) M. H. Brown and R. Sedgewick. A system for algorithm animation. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 177–186, New York, NY, USA, 1984. ACM Press.
- 16) S. R. Hansen, N. H. Narayanan, and D. Schrimpscher. Helping learners visualize and comprehend algorithms. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 2, 2000.
- 17) J. S. Gurka and W. Citrin. Testing effectiveness of algorithm animation. In *Proceedings, IEEE Symposium on Visual Languages*, pages 182–189, 1996.
- 18) M. D. Byrne, R. Catrambone, and J. T. Stasko. Do algorithm animations aid learning? Technical Report GIT-GVU-96-18, Georgia Institute of Technology, 1996.
- 19) C. Hundhausen and S. Douglas. Using visualizations to learn algorithms: should students construct their own, or view an expert's? In *Proceedings, IEEE Symposium on Visual Languages*, pages 21–28, 2000.
- 20) D. J. Jarc, M. B. Feldman, and R. S. Heller. Assessing the benefits of interactive prediction using web-based algorithm animation courseware. In *SIGCSE '00: Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*, pages 377–381, New York, NY, USA, 2000. ACM Press.
- 21) A. W. Lawrence, J. Stasko, and A. Badre. Empirically evaluating the use of animations to teach algorithms. In *Proceedings, IEEE Symposium on Visual Languages 1994*, pages 48–54. IEEE Computer Society, 1994.
- 22) Hansen, S., Narayanan, N. H., and Hegarty, M. (2002). Designing Educationally Effective Algorithm Visualizations. *Journal of Visual Languages & Computing*, 13(3), pages 291-317, 2002.

- 23) Khuri, S., "Designing Effective Algorithm Visualizations", *First International Program Visualization Workshop, Porvoo, Finland, University of Joensuu Press, pp. 1–12, 02/2001.*
- 24) Baker, A., and Milanovic, B. 2008. A Universal Extensible Architecture for Algorithm Visualization Systems. In *2008 International Conference on Computer Science and Software Engineering*, Wuhan, China December 12-14, pages 1-4, 2008.
- 25) P. Ihanola et al., "Taxonomy of effortless creation of algorithm visualizations", International Computing Education Research Workshop, *Proceedings of the 2005 International Workshop on Computing Education Research*, ACM, Seattle USA, October 2005, pp 123-133.
- 26) Michal Mnuk, "How helpful are systems for algorithm visualization?", Technical Report TUM-I0303, Institut für Informatik TU München, München Germany, April 2003.
- 27) R. Fleischer, L. Kučera, "Algorithm animation for teaching", *Lecture Notes In Computer Science*; Vol. 2269, Springer Verlag, London UK, 2001, pp. 113-128.
- 28) C.A. Shaffer, M. Cooper, and S.H. Edwards, "Algorithm visualization: a report on the state of the field", *ACM SIGCSE Bulletin*, Volume 39, Issue 1, Section: Algorithm Visualisation, ACM, New York USA, 2007, pp. 150-154.
- 29) Hundhausen et al., "A meta-study of algorithm visualization effectiveness", *Journal of Visual Languages & Computing*, Volume 13, Issue 3, June 2002, pp 259-290.
- 30) B. A. Price, R. M. Baecker, and I. S. Small, "A principled taxonomy of software visualization", *Journal of Visual Languages and Computing*, England, 1993, pp. 211-266.

- 31) Bingyao Jin; Mingmei Jin; Xiaoqing Xue, "Algorithm animation and its applications in instruction," *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on* , vol., no., pp.272-276, 5-6 July 2010.
- 32) Khuri, S. A user-centred approach for designing algorithm visualizations. *Informatik / Informatique, Special Issue on Visualization of Software* (Apr 2001), 12--16.
- 33) Jaime Urquiza-Fuentes and J. Angel Velazquez-Iturbide. 2009. A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems. *Trans. Comput. Educ.* 9, 2, Article 9 (June 2009), 21 pages.
- 34) Karavirta. V, "Facilitating algorithm animation creation and adoption in education", Licentiate's thesis, Helsinki University of Technology, available at <http://www.cs.hut.fi/Research/svg/publications/karavirta-lis.pdf>, 2007.
- 35) Colleen Kehoe, John Stasko, and Ashley Taylor, "Rethinking the evaluation of algorithm animations as learning aids: an observational study", *International Journal of Human Computer Studies*, Academic Press, Inc, Duluth, MN, USA, Feb. 2001, pp. 265-284.
- 36) M. D. Byrne, R. Catrambone, and J.T. Stasko, "Evaluating animations as student aids in learning computer algorithms", *Computer & Education*, 1999, pp. 253-278.
- 37) Tobias Lauer, "When does Algorithm Visualization Improve Algorithm Learning? Reviewing and Refining an Evaluation Framework", *Proceedings of informatics education Europe III*, Venice, Italy, 2008, pp. 265-284.
- 38) Ben-Bassat Levy, R., Ben-Ari, M. & Uronen, P.A. (2003). The Jeliot 2000 program animation system. *Computers & Education*, 40(1), 1-15.

- 39) Stasko, J., Badre, A., and Lewis, C. Do Algorithm Animations Assist Learning? An Empirical Study and Analysis. *Proceedings of ACM INTERCHI 1993 Conference on Human Factors in Computing Systems* (1993), 61-66.
- 40) Rajala, T., Laakso, M. J., Kaila, E., and Salakoski, T. 2008. Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. *Journal of Information Technology Education*, Volume 7, 16-32.
- 41) Korhonen, A., and Malmi, L. Algorithm Simulation with Automatic Assessment. *5th Annual ACM SIGCSE/SIGCUE Conference on Innovation and 152 Technology in Computer Science Education (ITiCSE 2000), Helsinki, Finland* (July 2000), 160-163.
- 42) Saraiya P, Shaffer CA, McCrickard DS, North C. Effective Features of Algorithm Visualizations. In: *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. Norfolk, VA: ACM; 2004. p. 382-6.
- 43) Naps, T., Rodger, S., Röbling, G., & Ross, R. (2006). Animation and visualization in the curriculum: opportunities, challenges, and successes. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education* (2006), pp. 328-329.
- 44) B. Price, R. Baecker, and I. Small. A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4:211-266, 1993.
- 45) Levy, R. B., & Ben-Ari, M. (2007). We work so hard and they don't use it: acceptance of software tools by teachers. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education* (2007), pp. 246-250.
- 46) C. Kann, R. Lindeman, and R. Heller. Integrating algorithm animation into a learning environment. *Computers & Education*, 28:223-228.

- 47) M.L. Cooper. Algorithm visualization: The state of the field. Master's thesis, Virginia Tech, April 2007.
- 48) Barbu, A., Dromowicz, M., Gao, X., Koester, M., and Wolf, C. Bubblesort-Animation, 1998. Available at <http://olli.informatik.uni-oldenburg.de/fpsort/Animation.html>
- 49) Guido Robling, Markus Schuer, and Bernd Freisleben. 2000. The ANIMAL algorithm animation tool. *SIGCSE Bull.* 32, 3 (July 2000), 37-40.
- 50) Thomas L. Naps, "JHAVE: Supporting Algorithm Visualization," *IEEE Computer Graphics and Applications*, pp. 49-55, September/October, 2005
- 51) Petre, M., & Green, T. (1993). Learning to read graphics: Some evidence that "seeing" an information display is an acquired skill. *Journal of Visual Languages and Computing*, 4(1), 55-70.
- 52) Petre, M. (1995). Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38(6), 33-44.
- 53) Stasko, J. (1998). Building software visualizations through direct manipulation and demonstration. In J. Stasko, J. Domingue, M. Brown, & B. Price (Eds.), *Software visualization - Programming as a multimedia experience* (pp. 187-203). Cambridge, MA: MIT Press.
- 54) Margarita Esponda-Argüero, 2008. Algorithmic Animation in Education - Review of Academic Experience. *J. Educational Computing Research*, Vol. 39(1) 1-15, 2008
- 55) Sedgewick, R. (2003). *Bundle of algorithms in Java: Fundamentals, data structures, sorting, searching, and graph algorithms* (3rd ed.), Reading, MA: Addison-Wesley.
- 56) Lohse, G., Biolski, K., Walker, N., & Rueter, H. (1994). A classification of visual representations. *Communications of the ACM*, 12(12), 36-49

- 57) Dondis, D. (1973). A primer of visual literacy. Cambridge, MA: MIT Press.
- 58) Goodrich, M., & Tamassia, R. (1998). Teaching the Analysis of Algorithms with Visual Proofs. Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education.
- 59) Scanlan, D. (1999). Structured flowcharts outperform pseudocode: An experimental comparison. *IEEE Software*, 6(5), 28-36
- 60) Knuth, D. (1984). Literate programming. *The Computer Journal*, 27(2), 97-111.
- 61) Glinert, E. (Ed.). (1990). Visual programming environments: Applications and issues. New York: IEEE Computer Society Press
- 62) Chang, S. (1987). Visual languages: A tutorial and survey. *IEEE Software*, 4(1), 29-39
- 63) Ciesielski, V., & McDonald, P. (2001). Using animation of state space algorithms to overcome student learning difficulties. In Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education (pp. 97-100). Canterbury, UK.
- 64) Stern, L., & Sterling, L. (1997). Teaching AI using animations reinforced by interactive exercises. Proceedings of the Second Australasian Conference on Computer Science Education (pp. 78-83). University of Melbourne, Australia
- 65) Jackson, D., & Fovargue, A. (1997). The use of animations to explain genetic algorithms. Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education (pp. 243-247). San Jose, CA

- 66) Selig, W., & Johannes, J. (1990). Reasoning visualization in expert systems: The applicability of algorithm animation techniques. Proceedings of the Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (Vol. 1, pp. 457-466). Charleston, SC
- 67) Domingue, J. (1998). Visualizing knowledge based systems. In J. Stasko, J. Domingue, M. Brown, & B. Price (Eds.), *Software visualization—Programming as a multimedia experience* (pp. 223-236). Cambridge, MA: MIT Press
- 68) Jackson, D., & Morton, I. (1996). Algorithm animation of neural networks. Proceedings of the 1st Conference on Integrating Technology into Computer Science Education (pp. 39-41). Barcelona, Spain
- 69) Hansen, S., Schrimsher, D., Narayanan N., & Hegarty, M. (1998). Empirical studies of animation-embedded hypermedia algorithm visualizations. Department of CS and Engineering, Auburn University. Technical Report CSE98-06
- 70) Stasko, J. (1997). Using student-built algorithm animations as learning aids. Proceedings of the 1997 ACM SIGCSE Conference (pp. 25-29). San Jose, CA
- 71) Hübscher-Younger, T., & Narayanan, N. (2003b). Dancing hamsters and marble statues: Characterizing student visualizations of algorithms. Proceedings of the 2003 ACM Symposium on Software Visualization (pp. 95-104). San Diego, CA
- 72) Faltin, N. (2002). *Strukturiertes aktives Lernen von Algorithmen mit interaktiven Visualisierungen*. Unpublished Ph.D. thesis. Computer Science Department. Oldenburg University
- 73) Rajat Anantharam, “Visualization of Software Engineering Diagrams Part – 1”, Utrecht University, available at <http://www.slidefinder.net/r/rajat1/10189299>, 2010

- 74) Brown, M., & Sedgewick, R. (1985). Techniques for Algorithm Animation. *Software, IEEE*, 2(1), 28-39
- 75) Doyle M., Rawe, B., and Rogers, A. 2007. *JDLX: Visualization of Dancing Links*, Northern Kentucky University, Highland Heights, KY
- 76) Building a Community and Establishing Best Practices in Algorithm Visualization through the AlgoViz Wiki. <http://research.cs.vt.edu/AVresearch/Documents/CCLI08.pdf>, May 2008.
- 77) Schweitzer, D., and Baird, L. 2006. The Design and Use of Interactive Visualization Applets for Teaching Ciphers. In *Proceedings of the 7th IEEE Workshop on Information Assurance, U.S. Military Academy*, West Point, NY, 21-23 June 2006
- 78) Alon, A. 2010. The AlgoViz Project: Building an Algorithm Visualization Web Community. Masters dissertation. Virginia Polytechnic Institute and State University, Virginia.
- 79) Ebel, G., & Ben-Ari, M. (2006). Affective effects of program visualization. In *Proceedings of the second international workshop on computing education research* (pp. 1-5). Canterbury, United Kingdom
- 80) Lauer, T. (2006). Learner interaction with algorithm visualizations: viewing vs. changing vs. constructing. In *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 202-206). Bologna, Italy
- 81) Algoviz, 2006. Algoviz wiki. <http://wiki.algoviz.org/AlgovizWiki/>
- 82) Maravic Cisar, S.; Pinter, R.; Radosav, D.; Cisar, P.; , "Software visualization: The educational tool to enhance student learning," *MIPRO, 2010 Proceedings of the 33rd International Convention* , vol., no., pp.990-994, 24-28 May 2010

- 83) Hundhausen, C. 2003. Exploring Cognitive, Social, and Cultural Dimensions of Visualization in Computer Science Education. In The Algorithms Studio Project, Washington State University, Pullman, WA
- 84) Hübscher-Younger, T., and N. H. Narayanan, "Constructive and collaborative learning of algorithms", *ACM SIGCSE Bulletin*, vol. 35, issue 1, pp. 6-10, 01/2003.
- 85) Virginia Tech Data Structures and Algorithm Visualization Research Group. Data Structures and Algorithm Visualization Wiki. <http://algoviz.cs.vt.edu>, 2008.
- 86) Hundhausen, C. D., and Douglas, S. A. 2002. Low-fidelity algorithm visualization. *Journal of Visual Languages and Computing* 13, 449–470.
- 87) Petri Ihantola, Ville Karavirta, Ari Korhonen, and Jussi Nikander. 2005. Taxonomy of effortless creation of algorithm visualizations. In *Proceedings of the first international workshop on Computing education research (ICER '05)*. ACM, New York, NY, USA, 123-133.
- 88) Guido Robling, Mike Joy, Andres Moreno, Atanas Radenski, Lauri Malmi, Andreas Kerren, Thomas Naps, Rockford J. Ross, Michael Clancy, Ari Korhonen, Rainer Oechsle, and J. Angel Velazquez Iturbide. 2008. Enhancing learning management systems to better support computer science education. *SIGCSE Bull.* 40, 4 (November 2008), 142-166.
- 89) Karavirta, V., Korhonen, A., Malmi, L. and Stalnacke, K. MatrixPro - A Tool for On The-Fly Demonstration of Data Structures and Algorithms. In Korhonen, A. Ed. *Proceedings of the Third Program Visualization Workshop*. (Coventry, UK). The University of Warwick, UK, 2004, 26-33.
- 90) Ari Korhonen. Visual Algorithm Simulation. Doctoral thesis, Helsinki University of Technology, 2003.

- 91) The AlgoViz Portal: Lowering the Barriers for Entry into an Online Educational Community. <http://research.cs.vt.edu/AVresearch/Documents/NSDL.pdf>, NSDL Annual meeting 2009.
- 92) Arnheim, R. (1969). *Visual thinking* (1st ed.). University of California Press; (2nd ed.). London: Faber and Faber.
- 93) Dunham, William (1974), *The Mathematical Universe*, John Wiley and Sons.
- 94) Ronald, B. Sorting Out Sorting: A Case Study of Software Visualization for Teaching Computer Science. In *Software Visualization: Programming as a Multimedia Experience*, chapter 24, Vol. 24 (1998), pp. 369-381.
- 95) Montgomery, L. N., Cross, J. H., Hendrix, T. D., & Barowski, L. A. (2008). Testing the jGRASP Structure Identifier with Data Structure Examples from Textbooks. In *Proceedings of the 46th ACM Southeast Conference* (pp. 198-203). Auburn, AL.

APPENDIX A

The complete data set that is collected for the analysis and research of this thesis based on the AlgoViz portal is listed below. Only part of this data is shown in the section 7.3, table 7.1. But the research conducted and the results calculated are based on the entire dataset.

1	A	B	C	D	E	F	G	H	I	J	K	L
2	SOFTWARE VISUALIZATION SYSTEM	ALGORITHM & DATA STRUCTURE VISUALIZATION	ALGOVIZ WEB LINK	1	2	3	4	5	6	7	8	9
3	Animal	Linked Lists	http://algoviz.org/catalog/entry/548	YES	NO	YES	NO	NO	NO	YES	YES	NO
4	jGrasp	Linked Lists	http://algoviz.org/catalog/entry/1828	YES	NO	YES	YES	YES	YES	YES	YES	YES
5	JHave	Binary Search Tree	http://algoviz.org/catalog/entry/895	YES	YES	YES	YES	YES	YES	YES	YES	NO
6	AlViE	Longest Common Subsequence	http://algoviz.org/catalog/entry/1921	YES	NO	YES	YES	YES	YES	YES	YES	YES
7	Trakla	AVL Tree	http://algoviz.org/node/1197	YES	NO	YES	YES	NO	NO	YES	YES	NO
8	Jeliot 3	Priority Queue	http://algoviz.org/catalog/entry/1824	YES	NO	YES	YES	YES	YES	YES	YES	NO
9	CS Animated	Red-Black Tree	http://algoviz.org/catalog/entry/864	YES	NO	YES	NO	NO	NO	YES	YES	NO
10	Algorithms In Action	Quick Sort	http://algoviz.org/catalog/entry/530	YES	NO	YES	YES	YES	YES	YES	YES	NO
11	Data Structures Navigator	Merge Sort	http://algoviz.org/catalog/entry/739	YES	NO	NO	NO	NO	YES	YES	YES	NO
12	Uuhistle	2,3,4 Tree	http://algoviz.org/catalog/entry/1880	YES	NO	YES	YES	YES	YES	YES	YES	YES
13	Data Structure Visualization	Dijkstra's Algorithm	http://algoviz.org/catalog/entry/749	YES	NO	NO	NO	YES	NO	YES	YES	NO
14	Interactive Data Structure Visualization	Graph Search	http://algoviz.org/catalog/entry/800	YES	NO	NO	NO	NO	NO	YES	YES	NO
15	Auckland	Huffman Coding	http://algoviz.org/catalog/entry/619	YES	NO	YES	YES	NO	YES	YES	YES	NO
16	Animator	Quick Sort	http://algoviz.org/catalog/entry/602	YES	NO	YES	YES	NO	YES	NO	NO	NO
17	Ghosh's AV	AVL Tree	http://algoviz.org/catalog/entry/640	YES	NO	NO	NO	YES	NO	NO	NO	NO
18	Sort Algorithm Animator	Insertion Sort	http://algoviz.org/catalog/entry/644	YES	NO	NO	NO	NO	NO	NO	NO	NO
19	Java Applets Centre	Binary Search Tree	http://algoviz.org/catalog/entry/685	NO	NO	YES	NO	YES	NO	NO	NO	NO
20	Galgo	Euler Path	http://algoviz.org/catalog/entry/772	YES	NO	NO	NO	YES	NO	NO	NO	NO
21	Gawain	Bubble Sort	http://algoviz.org/catalog/entry/775	YES	NO	YES	NO	NO	NO	NO	NO	NO
22	JCAT	Heap Sort	http://algoviz.org/catalog/entry/843	YES	NO	YES	YES	NO	YES	NO	NO	NO
23	JFLAP	Mealy Machine	http://algoviz.org/catalog/entry/848	YES	NO	NO	YES	YES	YES	YES	YES	NO
24	JAVENGA	Shortest Paths	http://algoviz.org/catalog/entry/878	YES	NO	YES	NO	NO	YES	NO	NO	NO
25	JAWAA	Shell Sort	http://algoviz.org/catalog/entry/889	YES	NO	NO	NO	NO	NO	YES	NO	NO
26	Kovac's Tree Project	B Tree	http://algoviz.org/catalog/entry/933	YES	NO	YES	NO	YES	NO	YES	NO	NO
27	OLLI	Heap Sort	http://algoviz.org/node/946	YES	NO	YES	YES	NO	YES	NO	NO	NO
28	Smith College AV	Sorting Algorithms	http://algoviz.org/node/1141	YES	NO	YES	YES	NO	YES	NO	NO	NO
29	Virginia Tech AV	Heap Sort	http://algoviz.org/node/1331	YES	NO	YES	YES	YES	YES	NO	NO	NO
30	Swan	Huffman Coding	http://algoviz.org/node/1174	YES	NO	NO	YES	YES	NO	NO	NO	NO
31	Upatras	Kruskal Algorithm	http://algoviz.org/node/1270	YES	NO	YES	NO	NO	NO	NO	NO	NO
32	UniSorter	Radix Sort	http://algoviz.org/node/1299	YES	NO	NO	NO	NO	NO	NO	NO	NO